NAG Library Function Document

nag_ml_mixed_regsn (g02jbc)

1 Purpose
nag_ml_mixed_regsn (g02jbc) fits a linear mixed effects regression model using maximum likelihood (ML).

2 Specification
#include <nag.h>
#include <nagg02.h>

void nag_ml_mixed_regsn (Integer n, Integer ncol, const double dat[],
Integer tddat, const Integer levels[], Integer yvid, Integer cwid,
Integer nfv, const Integer fvid[], Integer fint, Integer nrv,
const Integer rvid[], Integer nvpr, const Integer vpr[], Integer rint,
Integer svid, double gamma[], Integer *nff, Integer *nrf, Integer *df,
double *ml, Integer lb, double b[], double se[], Integer maxit,
double tol, Integer *warn, NagError *fail)

3 Description
nag_ml_mixed_regsn (g02jbc) 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

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

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 \( q \leq q \) groups with each group being identically distributed with expectations zero and variance \( \sigma^2_i \). The diagonal elements of matrix \( G \) therefore take one of the values \( \{ \sigma^2_i : i = 1, 2, \ldots, q \} \), 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_ml_mixed_regsn (g02jbc) 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_ml_mixed_regsn (g02jbc) 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_mixed_regsn (g02jbc) fits the model using a quasi-Newton algorithm to maximize the log-likelihood function:

\[-2l_R = \log(|V|) + (n)\log\left(r'V^{-1}r\right) + \log\left(2\pi/n\right)\]

where

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

Once the final estimates for \(\gamma^*\) have been obtained, the value of \(\sigma_R^2\) is given by:

\[\sigma_R^2 = \frac{(r'V^{-1}r)}{(n - p)}.\]

Case weights, \(W_c\), can be incorporated into the model by replacing \(X'X\) and \(Z'Z\) with \(X'W_cX\) and \(Z'W_cZ\) 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

1: \(n\) – Integer \(\text{Input}\)

\(On\ \text{entry:}\ n\), the number of observations.

\(Constraint:\ n \geq 1.\)

2: \(ncol\) – Integer \(\text{Input}\)

\(On\ \text{entry:}\ \text{the number of columns in the data matrix, DAT.}\)

\(Constraint:\ ncol \geq 1.\)

3: \(\text{dat}[n \times tddat]\) – const double \(\text{Input}\)

\(Note:\ \text{where DAT}(i, j)\ \text{appears in this document, it refers to the array element dat}[(i - 1) \times tddat + j - 1].\)

\(On\ \text{entry:}\ \text{array containing all of the data. For the ith observation:}\)

\(\text{DAT}(i, \text{yvid})\ \text{holds the dependent variable, y};\)

\(\text{if } \text{cwid} \neq 0, \text{DAT}(i, \text{cwid})\ \text{holds the case weights};\)

\(\text{if } \text{svid} \neq 0, \text{DAT}(i, \text{svid})\ \text{holds the subject variable.}\)

\(\text{The remaining columns hold the values of the independent variables.}\)

\(Constraint:\)

\(\text{if } \text{cwid} \neq 0, \text{DAT}(i, \text{cwid}) \geq 0.0;\)

\(\text{if } \text{levels}[j - 1] \neq 1, 1 \leq \text{DAT}(i, j) \leq \text{levels}[j - 1].\)
4: \texttt{tddat} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the stride separating matrix column elements in the array \texttt{dat}.
\textit{Constraint}: \( \texttt{tddat} \geq \texttt{ncol} \).

5: \texttt{levels[ncol]} – const Integer \hspace{1cm} \textit{Input}

\textit{On entry}: \texttt{levels[\(i - 1\)]} contains the number of levels associated with the \(i\)th variable of the data matrix \texttt{DAT}. If this variable is continuous or binary (i.e., only takes the values zero or one) then \texttt{levels[\(i - 1\)]} should be 1; if the variable is discrete then \texttt{levels[\(i - 1\)]} is the number of levels associated with it and \texttt{DAT(\(j, i\))} is assumed to take the values 1 to \texttt{levels[\(i - 1\)]}, for \(j = 1, 2, \ldots, \texttt{n}\).
\textit{Constraint}: \( \texttt{levels[\(i - 1\)]} \geq 1 \), for \( i = 1, 2, \ldots, \texttt{ncol} \).

6: \texttt{yvid} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the column of \texttt{DAT} holding the dependent, \texttt{y}, variable.
\textit{Constraint}: \( 1 \leq \texttt{yvid} \leq \texttt{ncol} \).

7: \texttt{cwid} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the column of \texttt{DAT} holding the case weights.
\text{If} \texttt{cwid} = 0, \text{no weights are used.}
\textit{Constraint}: \( 0 \leq \texttt{cwid} \leq \texttt{ncol} \).

8: \texttt{nfv} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the number of independent variables in the model which are to be treated as being fixed.
\textit{Constraint}: \( 0 \leq \texttt{nfv} < \texttt{ncol} \).

9: \texttt{fvid[nfv]} – const Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the columns of the data matrix \texttt{DAT} holding the fixed independent variables with \texttt{fvid[\(i - 1\)]} holding the column number corresponding to the \(i\)th fixed variable.
\textit{Constraint}: \( 1 \leq \texttt{fvid[\(i - 1\)]} \leq \texttt{ncol} \), for \( i = 1, 2, \ldots, \texttt{nfv} \).

10: \texttt{fint} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: flag indicating whether a fixed intercept is included (\texttt{fint} = 1).
\textit{Constraint}: \( \texttt{fint} = 0 \) or 1.

11: \texttt{nrv} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the number of independent variables in the model which are to be treated as being random.
\textit{Constraints}:
\begin{align*}
& 0 \leq \texttt{nrv} < \texttt{ncol}; \\
& \texttt{nrv} + \texttt{rint} > 0.
\end{align*}

12: \texttt{rvid[nrv]} – const Integer \hspace{1cm} \textit{Input}

\textit{On entry}: the columns of the data matrix \texttt{DAT} holding the random independent variables with \texttt{rvid[\(i - 1\)]} holding the column number corresponding to the \(i\)th random variable.
\textit{Constraint}: \( 1 \leq \texttt{rvid[\(i - 1\)]} \leq \texttt{ncol} \), for \( i = 1, 2, \ldots, \texttt{nrv} \).
13: \textbf{nvpr} – Integer \quad \textit{Input}

\textit{On entry:} if \texttt{rint} = 1 and \texttt{svid} \neq 0, \textbf{nvpr} is the number of variance components being estimated \(-2, (g-1)\), else \textbf{nvpr} = \(g\).

If \textbf{nvr} = 0, \textbf{nvpr} is not referenced.

\textit{Constraint:} if \textbf{nvr} \neq 0, 1 \leq \textbf{nvpr} \leq \textbf{nvr}.

14: \textbf{vpr[\textbf{nvrv}]} – const Integer \quad \textit{Input}

\textit{On entry:} \textbf{vpr}[i-1] holds a flag indicating the variance of the \(i\)th random variable. The variance of the \(i\)th random variable is \(\sigma_i^2\), where \(j = \textbf{vpr}[i-1] + 1\) if \texttt{rint} = 1 and \texttt{svid} \neq 0 and \(j = \textbf{vpr}[i-1]\) otherwise. Random variables with the same value of \(j\) are assumed to be taken from the same distribution.

\textit{Constraint:} \(1 \leq \textbf{vpr}[i-1] \leq \textbf{nvpr}\), for \(i = 1, 2, \ldots, \textbf{nvrv}\).

15: \textbf{rint} – Integer \quad \textit{Input}

\textit{On entry:} flag indicating whether a random intercept is included (\texttt{rint} = 1).

If \texttt{svid} = 0, \texttt{rint} is not referenced.

\textit{Constraint:} \texttt{rint} = 0 or 1.

16: \textbf{svid} – Integer \quad \textit{Input}

\textit{On entry:} the column of \texttt{DAT} holding the subject variable.

If \texttt{svid} = 0, no subject variable is used.

Specifying a subject variable is equivalent to specifying the interaction between that variable and all of the random-effects. Letting the notation \(Z_1 \times Z_S\) denote the interaction between variables \(Z_1\) and \(Z_S\), fitting a model with \texttt{rint} = 0, random-effects \(Z_1 + Z_2\) and subject variable \(Z_S\) is equivalent to fitting a model with random-effects \(Z_1 \times Z_S + Z_2 \times Z_S\) and no subject variable. If \texttt{rint} = 1 the model is equivalent to fitting \(Z_S + Z_1 \times Z_S + Z_2 \times Z_S\) and no subject variable.

\textit{Constraint:} \(0 \leq \texttt{svid} \leq \texttt{ncol}\).

17: \textbf{gamma[nvpr + 2]} – double \quad \textit{Input/Output}

\textit{On entry:} holds the initial values of the variance components, \(\gamma_0\), with \textbf{gamma}[i-1] the initial value for \(\sigma_i^2/\sigma_R^2\), for \(i = 1, 2, \ldots, g\). If \texttt{rint} = 1 and \texttt{svid} \neq 0, \texttt{g = nvpr} + 1, else \texttt{g = nvpr}.

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

\textit{On exit:} \textbf{gamma}[i-1], for \(i = 1, 2, \ldots, g\), holds the final estimate of \(\sigma_i^2\) and \textbf{gamma}[g] holds the final estimate for \(\sigma_R^2\).

\textit{Constraint:} \textbf{gamma}[0] = -1.0 or \textbf{gamma}[i-1] \geq 0.0, for \(i = 1, 2, \ldots, g\).

18: \textbf{nff} – Integer * \quad \textit{Output}

\textit{On exit:} the number of fixed effects estimated (i.e., the number of columns, \(p\), in the design matrix \(X\)).

19: \textbf{nrf} – Integer * \quad \textit{Output}

\textit{On exit:} the number of random effects estimated (i.e., the number of columns, \(q\), in the design matrix \(Z\)).

20: \textbf{df} – Integer * \quad \textit{Output}

\textit{On exit:} the degrees of freedom.
g02 – Correlation and Regression Analysis

21: **ml** – double *Output*

On exit: \(-2\lambda_R(\gamma)\) where \(\lambda_R\) is the log of the maximum likelihood calculated at \(\gamma\), the estimated variance components returned in **gamma**.

22: **lb** – Integer *Input*

On entry: the size of the array \(b\).

Constraint:

\[
ml \geq \text{fint} + \sum_{i=1}^{\text{nfv}} \max(\text{levels}[\text{fvid}[i-1]-1,1) + L_S \times \left(\text{rint} + \sum_{i=1}^{\text{nrv}} \text{levels}[\text{rvid}[i-1]-1]\right)
\]

where \(L_S = \text{levels}[\text{svid} - 1]\) if \(\text{svid} \neq 0\) and 1 otherwise.

23: **b[|b|]** – double *Output*

On exit: the parameter estimates, \((\beta, \nu)\), with the first \(\text{nff}\) elements of \(b\) containing the fixed effect parameter estimates, \(\beta\) and the next \(\text{nrf}\) elements of \(b\) containing the random effect parameter estimates, \(\nu\).

**Fixed effects**

If \(\text{fint} = 1\), \(b[0]\) contains the estimate of the fixed intercept. Let \(L_i\) denote the number of levels associated with the \(i\)th fixed variable, that is \(L_i = \text{levels}[\text{fvid}[i-1]-1]\). Define

\[
\begin{align*}
\text{fint} = 1 & \text{ if } F_i = 2 \\
\text{fint} = 0 & \text{ if } F_i = 1;
\end{align*}
\]

Then for \(i = 1, 2, \ldots, \text{nfv}\):

\[
\begin{align*}
& L_i > 1, b[F_i + j - 3] \text{ contains the parameter estimate for the } j\text{th level of the } i\text{th fixed variable, for } j = 2, 3, \ldots, L_i; \\
& L_i \leq 1, b[F_i - 1] \text{ contains the parameter estimate for the } i\text{th fixed variable.}
\end{align*}
\]

**Random effects**

Redefine \(L_i\) to denote the number of levels associated with the \(i\)th random variable, that is \(L_i = \text{levels}[\text{rvid}[i-1]-1]\). Define

\[
\begin{align*}
\text{rint} = 1 & \text{ if } R_i = 2 \\
\text{rint} = 0 & \text{ if } R_i = 1;
\end{align*}
\]

Then for \(i = 1, 2, \ldots, \text{nrv}\):

\[
\begin{align*}
& \text{svid} = 0, \\
& L_i > 1, b[\text{nff} + R_i + j - 2] \text{ contains the parameter estimate for the } j\text{th level of the } i\text{th random variable, for } j = 1, 2, \ldots, L_i; \\
& L_i \leq 1, b[\text{nff} + R_i - 1] \text{ contains the parameter estimate for the } i\text{th random variable;}
\end{align*}
\]

If \(\text{svid} \neq 0\),

let \(L_S\) denote the number of levels associated with the subject variable, that is \(L_S = \text{levels}[\text{svid} - 1]\):

\[
\begin{align*}
& L_i > 1, b[\text{nff} + (s - 1)L_S + R_i + j - 2] \text{ contains the parameter estimate for the interaction between the } s\text{th level of the subject variable and the } j\text{th level of the } i\text{th random variable, for } s = 1, 2, \ldots, L_S \text{ and } j = 1, 2, \ldots, L_i; \\
& L_i \leq 1, b[\text{nff} + (s - 1)L_S + R_i - 1] \text{ contains the parameter estimate for the interaction between the } s\text{th level of the subject variable and the } i\text{th random variable, for } s = 1, 2, \ldots, L_S; \\
& \text{rint} = 1, b[\text{nff}] \text{ contains the estimate of the random intercept.}
\end{align*}
\]
24: \( \text{se} | \text{lb} \) – double  
\( \text{Output} \)  
\textit{On exit}: the standard errors of the parameter estimates given in \( \text{b} \).

25: \( \text{maxit} \) – Integer  
\( \text{Input} \)  
\textit{On entry}: the maximum number of iterations.

If \( \text{maxit} < 0 \), the default value of 100 is used.

If \( \text{maxit} = 0 \), the parameter estimates \( (\beta, \nu) \) and corresponding standard errors are calculated based on the value of \( \gamma_0 \) supplied in \( \text{gamma} \).

26: \( \text{tol} \) – double  
\( \text{Input} \)  
\textit{On entry}: the tolerance used to assess convergence.

If \( \text{tol} \leq 0.0 \), the default value of \( 10^{-7} \) is used, where \( \epsilon \) is the \textit{machine precision}.

27: \( \text{warn} \) – Integer *  
\( \text{Output} \)  
\textit{On exit}: is set to 1 if a variance component was estimated to be a negative value during the fitting process. Otherwise \( \text{warn} \) is set to 0.

If \( \text{warn} = 1 \), the negative estimate is set to zero and the estimation process allowed to continue.

28: \( \text{fail} \) – NagError *  
\( \text{Input/Output} \)  
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

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

\textbf{NE_BAD_PARAM}
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.
On entry, invalid data: categorical variable with value greater than that specified in \( \text{levels} \).

\textbf{NE_CONV}
Routine failed to converge in \( \text{maxit} \) iterations: \( \text{maxit} = \langle \text{value} \rangle \).

\textbf{NE_FAIL_TOL}
Routine failed to converge to specified tolerance: \( \text{tol} = \langle \text{value} \rangle \).

\textbf{NE_INT}
On entry, \( \text{fint} = \langle \text{value} \rangle \).
Constraint: \( \text{fint} = 0 \) or 1.
On entry, \( \text{lb} \) too small: \( \text{lb} = \langle \text{value} \rangle \).
On entry, \( \text{levels}[I] < 1 \), for at least one \( I \).
On entry, \( n < 1 \) (nonzero weighted observations): \( n = \langle \text{value} \rangle \).
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 1 \).
On entry, \( \text{ncol} = \langle \text{value} \rangle \).
Constraint: \( 1 \leq \text{fvid}[i] \leq \text{ncol} \), for all \( i \).
On entry, \( ncol = \langle \text{value} \rangle \).
Constraint: \( 1 \leq rvid[i] \leq ncol \), for all \( i \).

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

On entry, \( nvpr = \langle \text{value} \rangle \).
Constraint: \( 1 \leq vpr[i] \leq nvpr \), for all \( i \).

On entry, \( rint = \langle \text{value} \rangle \).
Constraint: \( rint = 0 \) or \( 1 \).

**NE_INT_2**

On entry, \( cwid = \langle \text{value} \rangle \) and \( ncol = \langle \text{value} \rangle \).
Constraint: \( 0 \leq cwid \leq ncol \) and any supplied weights must be \( \geq 0.0 \).

On entry, \( nfv = \langle \text{value} \rangle \) and \( ncol = \langle \text{value} \rangle \).
Constraint: \( 0 \leq nfv < ncol \).

On entry, \( nrv = \langle \text{value} \rangle \) and \( ncol = \langle \text{value} \rangle \).
Constraint: \( 0 \leq nrv < ncol \) and \( nrv + rint > 0 \).

On entry, \( nvpr = \langle \text{value} \rangle \) and \( nr = \langle \text{value} \rangle \).
Constraint: \( 0 \leq nvpr \leq nr \) and \( nr \leq 0 \) or \( nvpr \geq 1 \).

On entry, \( svd = \langle \text{value} \rangle \) and \( ncol = \langle \text{value} \rangle \).
Constraint: \( 0 \leq svd \leq ncol \).

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

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

**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_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**

On entry, \( \text{gamma}[I] < 0 \), for at least one \( I \).

**NE_ZERO_DOF_ERROR**

Degrees of freedom \( < 1 \): \( \text{df} = \langle \text{value} \rangle \).

7 Accuracy

The accuracy of the results can be adjusted through the use of the \( \text{tol} \) argument.

8 Parallelism and Performance

\texttt{nag_ml_mixed_regsn (g02jbc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_ml_mixed_regsn (g02jbc) 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

Wherever possible any block structure present in the design matrix $Z$ should be modelled through a subject variable, specified via svid, rather than being explicitly entered into dat.

nag_ml_mixed_regsn (g02jbc) uses an iterative process to fit the specified model and for some problems this process may fail to converge (see fail.code = NE_CONV or NE_FAIL_TOL). If the function fails to converge then the maximum number of iterations (see maxit) or tolerance (see tol) may require increasing; try a different starting estimate in gamma. Alternatively, the model can be fit using restricted maximum likelihood (see nag_reml_mixed_regsn (g02jac)) or using the noniterative MIVQUE0.

To fit the model just using MIVQUE0, the first element of gamma should be set to −1 and maxit should be set to zero.

Although the quasi-Newton algorithm used in nag_ml_mixed_regsn (g02jbc) tends to require more iterations before converging compared to the Newton–Raphson algorithm recommended by Wolfinger et al. (1994), it does not require the second derivatives of the likelihood function to be calculated and consequently takes significantly less time per iteration.

10 Example

The following dataset is taken from Stroup (1989) and arises from a balanced split-plot design with the whole plots arranged in a randomized complete block-design.

In this example the full design matrix for the random independent variable, $Z$, is given by:

$$
Z = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$
where

\[
A = \begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

The block structure evident in (1) is modelled by specifying a four-level subject variable, taking the values \{1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4\}. The first column of 1s is added to \(A\) by setting \(\text{rint} = 1\). The remaining columns of \(A\) are specified by a three level factor, taking the values, \{1, 2, 3, 1, 2, 3, 1, \ldots\}.

### 10.1 Program Text

```c
/* nag_ml_mixed_regsn (g02jbc) Example Program. */
*
* Copyright 2014 Numerical Algorithms Group.
* *
* Mark 8, 2004. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>

int main(void)
{
    /* Scalars */
    double like, tol;
    Integer cwid, df, exit_status, fint, i, j, k, l, lb, maxit, n, ncol, nff,
        nfv;
    Integer nrf, nrv, nvpr, tddat, rint, svid, warnp, yvid, fnlevel, rnlevel;
    Integer lgamma, fl;
    /* Nag types */
    NagError fail;

    /* Arrays */
    double *b = 0, *dat = 0, *gamma = 0, *se = 0;
    Integer *fvid = 0, *levels = 0, *rvid = 0, *vpr = 0;

    // Define DAT(I, J) dat[(I-1)*tddat + J - 1]
    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_ml_mixed_regsn (g02jbc) Example Program Results\n\n");
    lb = 25;
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

    /* Read in the problem size information */
    #ifdef _WIN32
```

Mark 25
```c
scanf_s("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n] ",
        &n, &ncol, &nfv, &nr, &nvpr);
#else
scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n] ",
        &n, &ncol, &nfv, &nr, &nvpr);
#endif

/* Check problem size */
if (n < 0 || ncol < 0 || nfv < 0 || nr < 0 || nvpr < 0)
{
    printf("Invalid problem size, at least one of n, ncol, nfv, "
           "nr or nvpr is < 0\n";
    exit_status = 1;
    goto END;
}

/* Allocate memory first lot of memory */
if (!(levels = NAG_ALLOC(ncol, Integer)) ||
    !(fvid = NAG_ALLOC(nfv, Integer)) ||
    !(rvid = NAG_ALLOC(nr, Integer)) ||
    !(vpr = NAG_ALLOC(nvpr, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read in number of levels for each variable */
for (i = 1; i <= ncol; ++i)
{
#ifdef _WIN32
    scanf_s("%NAG_IFMT", &levels[i - 1]);
#else
    scanf("%NAG_IFMT", &levels[i - 1]);
#endif
}
#ifdef _WIN32
    scanf_s("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n] ", &svid,
        &cwid, &fint, &rint);
#else
    scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n] ", &svid,
        &cwid, &fint, &rint);
#endif
```

/* Read in the variance component flag */
for (i = 1; i <= nrv; ++i)
    {
#ifdef _WIN32
    scanf_s("%"NAG_IFMT", &vpr[i - 1]);
#else
    scanf("%"NAG_IFMT", &vpr[i - 1]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* If no subject specified, then ignore rint */
if (svid == 0)
    {
    rint = 0;
    }

/* Count the number of levels in the fixed parameters */
for (i = 1; fnlevel = 0; i <= nfv; ++i)
    {
    fl = levels[fvid[i - 1] - 1] - 1;
    fnlevel += (fl < 1)?1:fl;
    }
if (fint == 1)
    {
    fnlevel++;
    }

/* Count the number of levels in the random parameters */
for (i = 1, rnlevel = 0; i <= nrv; ++i)
    {
    rnlevel += levels[rvid[i - 1] - 1];
    }
if (rint)
    {
    rnlevel++;
    }

/* Calculate the sizes of the output arrays */
if (rint == 1)
    {
    lgamma = nvpr + 2;
    }
else
    {
    lgamma = nvpr + 1;
    }
if (svid)
    {
    lb = fnlevel + levels[svid-1] * rnlevel;
    }
else
    {
    lb = fnlevel + rnlevel;
    }
tddat = ncol;

/* Allocate remaining memory */
if (!((dat = NAG_ALLOC(n*ncol, double)) ||
    !(gamma = NAG_ALLOC(lgamma, double)) ||
    !(b = NAG_ALLOC(lb, double)) ||
    !(se = NAG_ALLOC(lb, double)))
    {
    printf("Allocation failure\n");
    }
exit_status = -1;
goto END;
}

/* Read in the Data matrix */
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= ncol; ++j)
    {
        #ifdef _WIN32
            scanf_s("%lf", &DAT(i, j));
        #else
            scanf("%lf", &DAT(i, j));
        #endif
    }
}

/* Read in the initial values for GAMMA */
for (i = 1; i < lgamma; ++i)
{
    #ifdef _WIN32
        scanf_s("%lf", &gamma[i - 1]);
    #else
        scanf("%lf", &gamma[i - 1]);
    #endif
}

/* Read in the maximum number of iterations */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n] ", &maxit);
#else
    scanf("%"NAG_IFMT"%*[\n] ", &maxit);
#endif

/* Run the analysis */
tol = 0.;
warnp = 0;
/* nag_ml_mixed_regsn (g02jbc).
 * Linear mixed effects regression using Maximum Likelihood
 * (ML)
 */
nag_ml_mixed_regsn(n, ncol, dat, tddat, levels, yvid, cwid, nfv, fvid, fint,
                   nrv, rvid, nvpr, vpr, rint, svid, gamma, &nff, &nrf, &df,
                   &like, lb, b, se, maxit, tol, &warnp, &fail);

/* Report the results */
if (fail.code == NE_NOERROR)
{
    /* Output results */
    if (warnp != 0)
    {
        printf("Warning: At least one variance component was ");
        printf("estimated to be negative and then reset to zero\n\n");
    }
    printf("Fixed effects (Estimate and Standard Deviation)\n\n");
    k = 1;
    if (fint == 1)
    {
        printf("Intercept\n\n%10.4f%10.4f\n", b[k - 1],
               se[k - 1]);
        ++k;
    }
    for (i = 1; i <= nfv; ++i)
    {
        for (j = 1; j <= levels[fvid[i - 1] - 1]; ++j)
        {
            if (levels[fvid[i - 1] - 1] != 1 && j == 1) continue;
            printf("Variable\n%"NAG_IFMT" Level%"NAG_IFMT"%10.4f%10.4f\n",
                   i, j, b[k - 1], se[k - 1]);
            ++k;
        }
    }
}
}
printf("\n");
printf("Random Effects (Estimate and Standard Deviation)\n");
if (svid == 0)
{
  for (i = 1; i <= nrv; ++i)
  {
    for (j = 1; j <= levels[rvid[i - 1] - 1]; ++j)
    {
      printf("%s%4"NAG_IFMT"%s%4"NAG_IFMT"%10.4f%10.4f\n",
        "Variable", i, " Level", j, b[k - 1], se[k - 1]);
      ++k;
    }
  }
}
else
{
  for (l = 1; l <= levels[svid - 1]; ++l)
  {
    if (rint == 1)
    {
      printf("%s%4"NAG_IFMT"%s%10.4f%10.4f\n",
        "Intercept for Subject Level", l, " ",
        b[k - 1], se[k - 1]);
      ++k;
    }
    for (i = 1; i <= nrv; ++i)
    {
      for (j = 1; j <= levels[rvid[i - 1] - 1]; ++j)
      {
        printf("%s%4"NAG_IFMT"%s%4"NAG_IFMT"%s%4"NAG_IFMT"
          "%10.4f%10.4f\n", "Subject Level", l, " ",
          " Variable", i, " Level", j, b[k - 1],
          se[k - 1]);
        ++k;
      }
    }
  }
}
printf("\n");
printf(" Variance Components\n");
for (i = 1; i <= nvpr + rint; ++i)
{
  printf("%4"NAG_IFMT"%10.4f\n", i, gamma[i - 1]);
}
printf("%s%10.4f\n", "SIGMA\^2 = ", gamma[nvpr + rint]);
printf("%s%10.4f\n", "-2LOG LIKE = ", like);
printf("%s"NAG_IFMT"
", "DF = ", df);
else
{
  printf("Routine nag_ml_mixed_regsn (g02jbc) failed, with error"
    " message:\n\n", fail.message);
}
END:
NAG_FREE(b);
NAG_FREE(dat);
NAG_FREE(gamma);
NAG_FREE(se);
NAG_FREE(fvid);
NAG_FREE(levels);
NAG_FREE(rvid);
NAG_FREE(vpr);
return exit_status;
10.2 Program Data

nag_ml_mixed_regrsn (g02jbc) Example Program Data
2 4 5 3 1 1
1 4 3 2 3
1 3 4 5 3 2 0 1 1
1
5 6 1 1 1 1
5 0 1 2 1 1
3 9 1 3 1 1
3 0 2 1 1 1
3 6 2 2 1 1
3 3 2 3 1 1
3 2 1 1 1 1
3 1 3 2 1 1
1 5 3 1 1 1
3 5 4 1 1 1
17 4 3 1 1
4 1 1 2 1
36 1 2 2 2
35 1 3 2 3
25 2 1 2 1
28 2 2 2 2
30 2 3 2 3
24 3 1 2 1
27 3 2 2 2
19 3 3 2 3
25 4 1 2 1
30 4 2 2 2
18 4 3 2 3
1.0 1.0
-1

10.3 Program Results

nag_ml_mixed_regrsn (g02jbc) Example Program Results

Fixed effects (Estimate and Standard Deviation)

Fixed effects (Estimate and Standard Deviation)

Variable 1 Level 2 1.0000 3.0461
Variable 1 Level 3 -11.0000 3.0461
Variable 2 Level 2 -8.2500 1.8736
Variable 3 Level 2 0.5000 2.6497
Variable 3 Level 3 7.7500 2.6497

Random Effects (Estimate and Standard Deviation)

Random Effects (Estimate and Standard Deviation)

Intercept for Subject Level 1

Intercept for Subject Level 2

Subject Level 1 Variable 1 Level 1 3.7276 2.6268
Subject Level 1 Variable 1 Level 2 -1.4476 2.6268
Subject Level 1 Variable 1 Level 3 0.3733 2.6268

Subject Level 2 Variable 1 Level 1 -3.7171 2.6268
Subject Level 2 Variable 1 Level 2 -1.2253 2.6268
Subject Level 2 Variable 1 Level 3 4.8125 2.6268

Subject Level 3 Variable 1 Level 1 0.5903 2.6268
Subject Level 3 Variable 1 Level 2 0.3987 2.6268
Subject Level 3 Variable 1 Level 3 -2.3806 2.6268

Subject Level 4 Variable 1 Level 1 -0.6009 2.6268
Subject Level 4 Variable 1 Level 2 2.2742 2.6268
Subject Level 4 Variable 1 Level 3 -2.8052 2.6268

Variance Components

Variance Components

1 46.7969
2 11.5365
SIGMA^2 = 7.0208
-2LOG LIKE = 141.6877
DF = 16