NAG C 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,

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

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

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 expectations zero and variance σ_i^2. The diagonal elements of matrix G therefore take one of the values $\{\sigma_i^2 : i = 1, \ldots, g\}$, 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 μ and a vector of $g+1$ variance components, γ, where $\gamma = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_{g-1}^2, \sigma_g^2, \sigma_R^2\}$. Rather than working directly with γ, nag_ml_mixed_regsn (g02jbc) uses an iterative process to estimate $\gamma^* = \{\sigma_1^2/\sigma_R^2, \sigma_2^2/\sigma_R^2, \ldots, \sigma_{g-1}^2/\sigma_R^2, \sigma_g^2/\sigma_R^2, 1\}$. Due to the iterative nature of the estimation a set of initial values, γ_0, for γ^* is required. nag_ml_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:

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

where

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

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

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

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

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

1: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: $\mathbf{n} \geq 1$.

2: **ncol** – Integer *Input*

On entry: the number of columns in the data matrix, **dat**.

Constraint: $\mathbf{ncol} \geq 2$.

3: **dat**[**tddat** × **n**] – const double *Input*

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

On entry: array containing all of the data. For the ith observation:

$\mathbf{DAT}(i, \mathbf{yvid})$ holds the dependent variable, y.

If $\mathbf{cwid} \neq 0$ then $\mathbf{DAT}(i, \mathbf{cwid})$ holds the case weights.

If $\mathbf{svid} \neq 0$ then $\mathbf{DAT}(i, \mathbf{svid})$ holds the subject variable.

The remaining columns hold the values of the independent variables.

Constraints:

if $\mathbf{cwid} \neq 0$, $\mathbf{DAT}(i, \mathbf{cwid}) \geq 0$;
if $\mathbf{levels}[j-1] \neq 1$, $\mathbf{DAT}(i,j) > 0$, $\mathbf{DAT}(i,j) \leq \mathbf{levels}[j-1]$.

4: **tddat** – Integer *Input*

On entry: the stride separating column elements in the array **dat**.

Constraint: $\mathbf{tddat} \geq \mathbf{n}$

5: **levels**[**ncol**] – const Integer *Input*

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

Constraint: **levels**$[i-1] \geq 1$, for $i = 1, 2, \ldots,$ **ncol**.

6: **yvid** – Integer *Input*

On entry: the column of **DAT** holding the dependent, y, variable.

Constraint: $1 \leq$ **yvid** $\leq$ **ncol**.

7: **cwid** – Integer *Input*

On entry: the column of **DAT** holding the case weights. If **cwid** $= 0$ then no weights are used.

Constraint: $0 \leq$ **cwid** $\leq$ **ncol**.

8: **nfv** – Integer *Input*

On entry: the number of independent variables in the model which are to be treated as being fixed.

Constraint: $0 \leq$ **nfv** $<$ **ncol**.

9: **fvid**[**nfv**] – const Integer *Input*

On entry: the columns of the data matrix **DAT** holding the fixed independent variables with **fvid**$[i-1]$ holding the column number corresponding to the ith fixed variable.

Constraint: $1 \leq$ **fvid**$[i-1] \leq$ **ncol**, for $i = 1, 2, \ldots,$ **nfv**.

10: **fint** – Integer *Input*

On entry: flag indicating whether a fixed intercept is included (**fint** $= 1$).

Constraint: **fint** $= 0$ or 1.

11: **nrv** – Integer *Input*

On entry: the number of independent variables in the model which are to be treated as being random.

Constraint: $0 \leq$ **nrv** $<$ **ncol**.

12: **rvid**[**nrv**] – const Integer *Input*

On entry: the columns of the data matrix **DAT** holding the random independent variables with **rvid**$[i-1]$ holding the column number corresponding to the ith random variable.

Constraint: $1 \leq$ **rvid**$[i-1] \leq$ **ncol**, for $i = 1, 2, \ldots,$ **nrv**.

13: **nvpr** – Integer *Input*

On entry: if **rint** $= 1$ and **svid** $\neq 0$, **nvpr** is the number of variance components being estimated $-$ 2, $(g-1)$, else **nvpr** $= g$. If **nrv** $= 0$, **nvpr** is not referenced.

Constraint: if **nrv** $\neq 0$, $1 \leq$ **nvpr** $\leq$ **nrv**.

14: **vpr**[**nrv**] – const Integer *Input*

On entry: **vpr**$[i-1]$ holds a flag indicating the variance of the ith random variable. The variance of the ith random variable is σ_j^2, where $j =$ **vpr**$[i-1] + 1$ if **rint** $= 0$ and **svid** $\neq 0$ and $j =$ **vpr**$[i-1]$ otherwise. Random variables with the same value of j are assumed to be taken from the same distribution.

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

15: **rint** – Integer *Input*

On entry: flag indicating whether a random intercept is included (**rint** $= 1$). If **svid** $= 0$ **rint** is not referenced.

Constraint: **rint** $= 0$ or 1.

16: **svid** – Integer *Input*

On entry: the column of **DAT** holding the subject variable. If **svid** $= 0$ then 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 **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 **rint** $= 1$ the model is equivalent to fitting $Z_S + Z_1 \times Z_S + Z_2 \times Z_S$ and no subject variable.

Constraint: $0 \leq$ **svid** $\leq$ **ncol**.

17: **gamma**[**nvpr** + **2**] – 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, $i = 1, 2, \ldots, g$. If **rint** $= 1$ and **svid** $\neq 0$ then $g =$ **nvpr** $+ 1$, else $g =$ **nvpr**.

If **gamma**$[0] = -1$ then 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 $i = 1, 2, \ldots, g$ holds the final estimate of σ_i^2 and **gamma**$[g]$ holds the final estimate for σ_R^2.

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

18: **nff** – Integer * *Output*

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

19: **nrf** – Integer * *Output*

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

20: **df** – Integer * *Output*

On exit: the degrees of freedom.

21: **ml** – 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**.

22: **lb** – Integer *Input*

On entry: the size of the array **b**.

Constraint: $\mathbf{lb} \geq \mathbf{fint} + \sum_{i=1}^{\mathbf{nfv}} \max(\mathbf{levels}[\mathbf{fvid}[i-1]] - 1, 1) +$

$L_S \times \left(\mathbf{rint} + \sum_{i=1}^{\mathbf{nrv}} \mathbf{levels}[\mathbf{rvid}[i-1]] \right)$ where $L_S = \mathbf{levels}[\mathbf{svid} - 1]$ if **svid** $\neq 0$ and 1 otherwise

23: **b**[**lb**] – double *Output*

On exit: the argument estimates, (β, ν), with the first **nff** elements of **b** containing the fixed effect argument estimates, β and the next **nrf** elements of **b** containing the random effect argument estimates, ν.

Fixed effects

If **fint** $= 1$, **b**$[0]$ contains the estimate of the fixed intercept. Let L_i denote the number of levels associated with the ith fixed variable, that is $L_i =$ **levels**[**fvid**$[i-1]-1]$. Define

if **fint** $= 1$, $F_1 = 2$ else if **fint** $= 0$, $F_1 = 1$;

$F_{i+1} = F_i + \max(L_i - 1, 1)$, $i \geq 1$.

Then for $i = 1, 2, \ldots,$ **nfv**:

if $L_i > 1$, **b**$[F_i + j - 3]$ contains the argument estimate for the jth level of the ith fixed variable, $j = 2, 3, \ldots, L_i$;
if $L_i \leq 1$, **b**$[F_i - 1]$ contains the argument estimate for the ith fixed variable.

Random effects

Redefining L_i to denote the number of levels associated with the ith random variable, that is $L_i =$ **levels**[**rvid**$[i-1]-1]$. Define

if **rint** $= 1$, $R_1 = 2$ else if **rint** $= 0$, $R_1 = 1$;
$R_{i+1} = R_i + L_i$, $i \geq 1$.

Then for $i = 1, 2, \ldots,$ **nrv**:

if **svid** $= 0$,

if $L_i > 1$, **b**[**nff** $+ R_i + j - 2$] contains the argument estimate for the jth level of the ith random variable, $j = 1, 2, \ldots, L_i$;
if $L_i \leq 1$, **b**[**nff** $+ R_i - 1$] contains the argument estimate for the ith random variable;

if **svid** $\neq 0$,

let L_S denote the number of levels associated with the subject variable, that is $L_S =$ **levels**[**svid** $- 1$];
if $L_i > 1$, **b**[**nff** $+ (s-1)L_S + R_i + j - 2$] contains the argument estimate for the interaction between the sth level of the subject variable and the jth level of the ith random variable, $s = 1, 2, \ldots, L_S$ and $j = 1, 2, \ldots, L_i$;
if $L_i \leq 1$, **b**[**nff** $+ (s-1)L_S + R_i - 1$] contains the argument estimate for the interaction between the sth level of the subject variable and the ith random variable, $s = 1, 2, \ldots, L_S$;
if **rint** $= 1$, **b**(**nff** $+ 1$) contains the estimate of the random intercept.

24: **se**[**lb**] – double *Output*

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

25: **maxit** – Integer *Input*

On entry: the maximum number of iterations.

maxit < 0

The default value of 100 is used.

maxit $= 0$

The argument estimates (β, μ) and corresponding standard errors are calculated based on the value of γ_0 supplied in **gamma**.

26: **tol** – double *Input*

On entry: the tolerance used to assess convergence. If **tol** $= 0$ then the default value of $\epsilon^{0.7}$ is used, where ϵ is the ***machine precision***.

27: **warn** – Integer * *Output*

On exit: is set to 1 if a variance component was estimated to be a negative value during the fitting process. Otherwise **warn** is set to 0.

If **warn** $= 1$, the negative estimate is set to zero and the estimation process allowed to continue.

28: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, invalid data: categorical variable with value greater than that specified in **levels**.

NE_CONV

Routine failed to converge in **maxit** iterations: **maxit** $= \langle value \rangle$.

NE_FAIL_TOL

Routine failed to converge to specified tolerance: **tol** $= \langle value \rangle$.

NE_INT

On entry, **fint** $\neq 0$ and **fint** $\neq 1$: **fint** $= \langle value \rangle$.

On entry, **fvid**$[i] < 1$ or **fvid**$[i] >$ **ncol**, for at least one i : **ncol** $= \langle value \rangle$.

On entry, **lb** too small: **lb** $= \langle value \rangle$.

On entry, **levels**$[i] < 1$, for at least one i.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

On entry, **ncol** $= \langle value \rangle$.
Constraint: **ncol** ≥ 2.

On entry, **rint** $\neq 0$ and **rint** $\neq 1$: **rint** $= \langle value \rangle$.

On entry, **rvid**$[i] < 1$ or **rvid**$[i] >$ **ncol**, for at least one i : **ncol** $= \langle value \rangle$.

On entry, **tddat** $= \langle value \rangle$.
Constraint: **tddat** > 0.

On entry, **vpr**$[i] < 1$ or **vpr**$[i] >$ **nvpr**, for at least one i : **nvpr** $= \langle value \rangle$.

NE_INT_2

On entry, **cwid** < 0 or **cwid** $>$ **ncol** or -ve weight: **cwid** $= \langle value \rangle$, **ncol** $= \langle value \rangle$.

On entry, **nfv** < 0 or **nfv** $\geq$ **ncol**: **nfv** $= \langle value \rangle$, **ncol** $= \langle value \rangle$.

On entry, **nrv** < 0 or **nrv** $\geq$ **ncol**: **nrv** $= \langle value \rangle$, **ncol** $= \langle value \rangle$.

On entry, **nvpr** < 0 or **nvpr** $>$ **nrv** or (**nrv** > 0 and **nvpr** < 1): **nvpr** $= \langle value \rangle$, **nrv** $= \langle value \rangle$.

On entry, **svid** < 0 or **svid** $>$ **ncol**: **svid** $= \langle value \rangle$, **ncol** $= \langle value \rangle$.

On entry, **tddat** $<$ **n** : **tddat** $= \langle value \rangle$, **n** $= \langle value \rangle$.

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

On entry, **yvid** < 1 or **yvid** $>$ **ncol**: **yvid** $= \langle value \rangle$, **ncol** $= \langle value \rangle$.

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 consult NAG for assistance.

NE_REAL

On entry, **gamma**$[i] < 0$, for at least one i.

NE_ZERO_DOF_ERROR

Degrees of freedom < 1: **df** $= \langle value \rangle$.

7 Accuracy

The accuracy of the results can be adjusted through the use of the **tol** argument.

8 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_FAIL_TOL**). If the function fails to converge then the maximum number of iterations (see **maxit**) or tolerance (see **tol**) may require increasing. Alternatively, the model can be fit using restricted maximum likelihood (see nag_reml_mixed_regsn (g02jac)) or using the non-iterative 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 consequentially takes significantly less time per iteration.

9 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:

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

$$= \begin{pmatrix} A&0&0&0\\ 0&A&0&0\\ 0&0&A&0\\ 0&0&0&A\\ A&0&0&0\\ 0&A&0&0\\ 0&0&A&0\\ 0&0&0&A \end{pmatrix}, \qquad (1)$$

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 **rint** $= 1$. The remaining columns of A are specified by a three level factor, taking the values, $\{1,2,3,1,2,3,1,\ldots\}$.

9.1 Program Text

```
/* nag_ml_mixed_regsn (g02jbc) Example Program.
 *
 * Copyright 2004 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, nv, 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);
  Vprintf("nag_ml_mixed_regsn (g02jbc) Example Program Results\n\n");
  lb = 25;
  /* Skip heading in data file */
  Vscanf("%*[^\n] ");

  /* Read in the problem size information */
  Vscanf("%ld%ld%ld%ld%ld%*[^\n] ", &n,
         &ncol, &nfv, &nrv, &nvpr);
  nv = nfv + nrv;

  /* Check problem size */
  if (n < 0 || ncol < 0 || nfv < 0 || nrv < 0 || nvpr < 0)
    {
      Vprintf("Invalid problem size, at least one of n, ncol, nfv, nrv 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(nrv,Integer)) ||
       !(vpr = NAG_ALLOC(nrv, Integer)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read in number of levels for each variable */
  for (i = 1; i <= ncol; ++i)
    {
      Vscanf("%ld", &levels[i - 1]);
    }
  Vscanf("%*[^\n] ");

  /* Read in model information */
  Vscanf("%ld", &yvid);
  for (i = 1; i <= nfv; ++i)
    {
      Vscanf("%ld", &fvid[i - 1]);
    }
  for (i = 1; i <= nrv; i++)
    {
      Vscanf("%ld", &rvid[i - 1]);
    }
  Vscanf("%ld%ld%ld%ld%*[^\n] ", &svid, &cwid,
         &fint, &rint);

  /* Read in the variance component flag */
  for (i = 1; i <= nrv; ++i)
    {
```

```
    Vscanf("%ld", &vpr[i - 1]);
  }
Vscanf("%*[^\n] ");

/* 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)))
  {
    Vprintf("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)
      {
        Vscanf("%lf",&DAT(i,j));
      }
  }
```

```
  /* Read in the initial values for GAMMA */
  for (i = 1; i < lgamma; ++i)
    {
      Vscanf("%lf", &gamma[i - 1]);
    }

  /* Read in the maximum number of iterations */
  Vscanf("%ld%*[^\n] ",   &maxit);

  /* 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)
        {
          Vprintf("%s", "Warning: At least one variance component was ");
          Vprintf("%s", "estimated to be negative and then reset to zero");
          Vprintf("\n");
        }
      Vprintf("%s\n\n", "Fixed effects (Estimate and Standard Deviation)");
      k = 1;
      if (fint == 1)
        {
          Vprintf("%s%10.4f%10.4f\n",   "Intercept             ",  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;
              Vprintf("%s%4ld%s%4ld%10.4f%10.4f\n", "Variable",
                      i,  " Level",  j,  b[k - 1],  se[k - 1]);
              ++k;
            }
        }

      Vprintf("\n");
      Vprintf("%s%s\n", "Random Effects (Estimate and Standard", " Deviation)");
      if (svid == 0)
        {
          for (i = 1; i <= nrv; ++i)
            {
              for (j = 1; j <= levels[rvid[i - 1] - 1]; ++j)
                {
                  Vprintf("%s%4ld%s%4ld%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)
                {
                  Vprintf("%s%4ld%s%10.4f%10.4f\n",
```

```
                         "Intercept for Subject Level", 1, "         ",
                         b[k - 1],  se[k - 1]);
                 ++k;
               }
             for (i = 1; i <= nrv; ++i)
               {
                 for (j = 1; j <= levels[rvid[i - 1] - 1]; ++j)
                   {
                     Vprintf("%s%4ld%s%4ld%s%4ld"
                             "%10.4f%10.4f\n", "Subject Level", 1, " Variable",
                             i, " Level", j, b[k - 1], se[k - 1]);
                     ++k;
                   }
               }
           }
       }

      Vprintf("\n");
      Vprintf("%s\n",   " Variance Components");
      for (i = 1; i <= nvpr + rint; ++i)
        {
          Vprintf("%4ld%10.4f\n",   i,  gamma[i - 1]);
        }
      Vprintf("%s%10.4f\n\n",   "SIGMAˆ2     = ",  gamma[nvpr + rint]);

      Vprintf("%s%10.4f\n\n",   "-2LOG LIKE  = ",  like);
      Vprintf("%s%ld\n",   "DF          = ",  df);
    }
  else
    {
      Vprintf("Routine nag_ml_mixed_regsn (g02jbc) failed, with error message:"
              "\n%s\n", fail.message);
    }

 END:
  if (b) NAG_FREE(b);
  if (dat) NAG_FREE(dat);
  if (gamma) NAG_FREE(gamma);
  if (se) NAG_FREE(se);
  if (fvid) NAG_FREE(fvid);
  if (levels) NAG_FREE(levels);
  if (rvid) NAG_FREE(rvid);
  if (vpr) NAG_FREE(vpr);
  return exit_status;
}
```

9.2 Program Data

```
nag_ml_mixed_regsn (g02jbc) Example Program Data
24 5 3 1 1
1 4 3 2 3
1 3 4 5 3 2 0 1 1
1
56  1  1  1  1
50  1  2  1  1
39  1  3  1  1
30  2  1  1  1
36  2  2  1  1
33  2  3  1  1
32  3  1  1  1
31  3  2  1  1
15  3  3  1  1
30  4  1  1  1
35  4  2  1  1
17  4  3  1  1
41  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
```

9.3 Program Results

```
nag_ml_mixed_regsn (g02jbc) Example Program Results

Fixed effects (Estimate and Standard Deviation)

Intercept                  37.0000    4.0421
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)
Intercept for Subject Level   1            10.7631    3.8855
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
Intercept for Subject Level   2            -0.5269    3.8855
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
Intercept for Subject Level   3            -5.6450    3.8855
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
Intercept for Subject Level   4            -4.5912    3.8855
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
   1   46.7969
   2   11.5365
SIGMA^2     =     7.0208

-2LOG LIKE  =   141.6877

DF          = 16
```