nag_mv_orthomax (g03bac)

1. Purpose

nag_mv_orthomax (g03bac) computes orthogonal rotations for a matrix of loadings using a generalized orthomax criterion.

2. Specification

```c
#include <nag.h>
#include <nagg03.h>

void nag_mv_orthomax(Nag_RotationLoading stand, double g, Integer nvar,
                      Integer k, double fl[], Integer tdf, double flr[], double r[],
                      Integer tdr, double acc, Integer maxit, Integer *iter, NagError *fail)
```

3. Description

Let Λ be the \( p \times k \) matrix of loadings from a variable-directed multivariate method, e.g., canonical variate analysis or factor analysis. This matrix represents the relationship between the original \( p \) variables and the \( k \) orthogonal linear combinations of these variables, the canonical variates or factors. The latter are only unique up to a rotation in the \( k \)-dimensional space they define. A rotation can then be found that simplifies the structure of the matrix of loadings, and hence the relationship between the original and the derived variables. That is, the elements, \( \lambda_{ij}^* \), of the rotated matrix, \( \Lambda^* \), are either relatively large or small. The rotations may be found by minimizing the criterion:

\[
V = \sum_{j=1}^{k} \sum_{i=1}^{p} (\lambda_{ij}^*)^4 - \frac{\gamma}{p} \sum_{j=1}^{k} \left( \sum_{i=1}^{p} (\lambda_{ij}^*)^2 \right)^2
\]

where the constant \( \gamma \) gives a family of rotations with \( \gamma = 1 \) giving varimax rotations and \( \gamma = 0 \) giving quartimax rotations.

It is generally advised that factor loadings should be standardised, so that the sum of squared elements for each row is one, before computing the rotations.

The matrix of rotations, \( R \), such that \( \Lambda^* = \Lambda R \), is computed using first an algorithm based on that described by Cooley and Lohnes (1971), which involves the pairwise rotation of the factors. Then a final refinement is made using a method similar to that described by Lawley and Maxwell (1971), but instead of the eigenvalue decomposition, the algorithm has been adapted to incorporate a singular value decomposition.

4. Parameters

**stand**

Input: indicates if the matrix of loadings is to be row standardised before rotation.

- If **stand** = Nag_RoLoadStand the loadings are row standardised.
- If **stand** = Nag_RoLoadNotStand the loadings are left unstandardised.

Constraint: **stand** = Nag_RoLoadStand or Nag_RoLoadNotStand.

**g**

Input: the criterion constant, \( \gamma \), with \( \gamma = 1.0 \) giving varimax rotations and \( \gamma = 0.0 \) giving quartimax rotations.

Constraint: **g** \( \geq 0.0 \).

**nvar**

Input: The number of original variables, \( p \).

Constraint: **nvar** \( \geq k \).

**k**

Input: The number of derived variates or factors, \( k \).

Constraint: **k** \( \geq 2 \).
nag_mv_orthomax

The matrix of loadings, Λ. fl[i - 1][j - 1] must contain the loading for the ith variable on the jth factor, for i = 1, 2, …, p; j = 1, 2, …, k.

Output: if stand = Nag_RoLoadStand the elements of fl are standardised so that the sum of squared elements for each row is 1.0 and then after, the computation of the rotations are rescaled; this may lead to slight differences between the input and output values of fl. If stand = Nag_RoLoadNotStand, fl will be unchanged on exit.

tdf
Input: the last dimension of the arrays fl and flr as declared in the calling program.
Constraint: tdf ≥ k.

flr[nvar][tdf]
Output: the rotated matrix of loadings, Λ*. flr[i - 1][j - 1] will contain the rotated loading for the ith variable on the jth factor, for i = 1, 2, …, p; j = 1, 2, …, k.

r[k][tdr]
Output: the matrix of rotations, R.

tdr
Input: the last dimension of the array r as declared in the calling program.
Constraint: tdr ≥ k.

acc
Input: indicates the accuracy required. The iterative procedure of Cooley and Lohnes (1971) will be stopped and the final refinement computed when the change in V is less than acc × max(1.0, V). If acc is greater than or equal to 0.0 but less than machine precision, or if acc is greater than 1.0, then machine precision will be used instead. It is suggested that acc be set to 0.00001.
Constraint: acc ≥ 0.0.

maxit
Input: the maximum number of iterations. It is suggested that maxit be set to 30.
Constraint: maxit ≥ 1.

iter
Output: the number of iterations performed.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_BAD_PARAM
On entry, parameter stand had an illegal value.

NE_INT_ARG_LT
On entry, k must not be less than 2: k = ⟨value⟩.

NE_REAL_ARG_LT
On entry, g must not be less than 0.0: g = ⟨value⟩.
On entry, acc must not be less than 0.0: acc = ⟨value⟩.
NE_SVD_NOT_CONV
The singular value decomposition has failed to converge.
This is an unlikely error exit.

NE_ACC_ITER
The algorithm to find R has failed to reach the required accuracy in the given number of
iterations, ⟨value⟩. Try increasing acc or increasing maxit. The returned solution should be
a reasonable approximation.

NE_ALLOC_FAIL
Memory allocation failed.

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.

6. Further Comments
If the results of a principal component analysis as carried out by nag_mv_prin_comp (g03aac) are to
be rotated, the loadings as returned in the array p by nag_mv_prin_comp (g03aac) can be supplied
via the parameter fl to nag_mv_orthomax. The resulting rotation matrix can then be used to rotate
the principal component scores as returned in the array v by nag_mv_prin_comp (g03aac). The
routine dgemm (f06yac) may be used for this matrix multiplication.

6.1. Accuracy
The accuracy is determined by the value of acc.

6.2. References
Lawley D N and Maxwell A E (1971) Factor Analysis as a Statistical Method Butterworths (2nd
Edition).

7. See Also
nag_mv_prin_comp (g03aac)
dgemm (f06yac)

8. Example
The example is taken from page 75 of Lawley and Maxwell (1971). The results from a factor
analysis of ten variables using three factors are input and rotated using varimax rotations without
standardising rows.

8.1. Program Text
/* nag_mv_orthomax (g03bac) Example Program.

* Copyright 1998 Numerical Algo...
double r[MMAX][MMAX];
double fl[NMAX][MMAX], flr[NMAX][MMAX];

Integer iter, nvar;
Integer i, j, k;
Integer maxit;
Integer tdf = MMAX;
Integer tdr = MMAX;

char char_stand[2];
Nag_RotationLoading stand;

Vprintf("g03bac Example Program Results\n\n");
/* Skip heading in data file */
Vscanf("%*[\n"]);
Vscanf("%ld", &nvar);
Vscanf("%ld", &k);
Vscanf("%lf", &g);
Vscanf("%s", char_stand);
Vscanf("%lf", &acc);
Vscanf("%ld", &maxit);
if (*char_stand == 'S')
    stand = Nag_RoLoadStand;
else
    stand = Nag_RoLoadNotStand;
if (nvar <= NMAX && k <= MMAX)
{
    for (i = 0; i < nvar; ++i)
    {
        for (j = 0; j < k; ++j)
            Vscanf("%lf", &fl[i][j]);
    }
g03bac(stand, g, nvar, k, (double *)fl, tdf, (double *)flr, (double *)r,
        tdr, acc, maxit, &iter, NAGERR_DEFAULT);
    Vprintf("\n Rotated factor loadings\n\n");
    for (i = 0; i < nvar; ++i)
    {
        for (j = 0; j < k; ++j)
            Vprintf(" %8.3f", flr[i][j]);
        Vprintf("\n");
    }
    Vprintf("\n Rotation matrix\n\n");
    for (i = 0; i < k; ++i)
    {
        for (j = 0; j < k; ++j)
            Vprintf(" %8.3f", r[i][j]);
        Vprintf("\n");
    }
    exit(EXIT_SUCCESS);
}
else
{
    Vprintf("Incorrect input value of nvar or k.\n");
    exit(EXIT_FAILURE);
}
}

8.2. Program Data

g03bac Example Program Data
10 3 1.0 0.00001 20
0.788  -0.152  -0.352
0.874  0.381  0.041
0.814  -0.043  -0.213
0.798  -0.170  -0.204
0.641  0.070  -0.042
8.3. Program Results

g03bac Example Program Results

Rotated factor loadings

\[
\begin{array}{ccc}
0.329 & -0.289 & -0.759 \\
0.849 & -0.273 & -0.340 \\
0.450 & -0.327 & -0.633 \\
0.345 & -0.397 & -0.657 \\
0.453 & -0.276 & -0.370 \\
0.263 & -0.615 & -0.464 \\
0.332 & -0.561 & -0.485 \\
0.472 & -0.684 & -0.183 \\
0.209 & -0.754 & -0.354 \\
0.423 & -0.514 & -0.409 \\
\end{array}
\]

Rotation matrix

\[
\begin{array}{ccc}
0.633 & -0.534 & -0.560 \\
0.758 & 0.573 & 0.311 \\
0.155 & -0.622 & 0.768 \\
\end{array}
\]