nag_mv_factor (g03cac)

1. Purpose

nag_mv_factor (g03cac) computes the maximum likelihood estimates of the parameters of a factor analysis model. Either the data matrix or a correlation/covariance matrix may be input. Factor loadings, communalities and residual correlations are returned.

2. Specification

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

void nag_mv_factor(Nag_FacMat matrix, Integer n, Integer m,
            double x[], Integer tdx, Integer nvar, Integer isx[],
            Integer nfac,  double wt[], double e[], double stat[],
            double com[], double psi[], double res[],
            double fl[], Integer tdfl, Nag_E04_Opt *options,
            double eps, NagError *fail)
```

3. Description

Let p variables, $x_1, x_2, \ldots, x_p$, with variance-covariance matrix Σ be observed. The aim of factor analysis is to account for the covariances in these p variables in terms of a smaller number, k, of hypothetical variables, or factors, $f_1, f_2, \ldots, f_k$. These are assumed to be independent and to have unit variance. The relationship between the observed variables and the factors is given by the model:

$$x_i = \sum_{j=1}^{k} \lambda_{ij} f_j + e_i \qquad i = 1, 2, \ldots, p$$

where λ_{ij}, for $i = 1, 2, \ldots, p$; $j = 1, 2, \ldots, k$, are the factor loadings and e_i, for $i = 1, 2, \ldots, p$, are independent random variables with variances ψ_i, for $i = 1, 2, \ldots, p$. The ψ_i represent the unique component of the variation of each observed variable. The proportion of variation for each variable accounted for by the factors is known as the communality. For this routine it is assumed that both the k factors and the e_i's follow independent Normal distributions.

The model for the variance-covariance matrix, Σ, can be written as:

$$\Sigma = \Lambda\Lambda^T + \Psi \tag{1}$$

where Λ is the matrix of the factor loadings, λ_{ij}, and Ψ is a diagonal matrix of unique variances, ψ_i, for $i = 1, 2, \ldots, p$.

The estimation of the parameters of the model, Λ and Ψ, by maximum likelihood is described by Lawley and Maxwell (1971). The log likelihood is:

$$-\frac{1}{2}(n-1)\log(|\Sigma|) - \frac{1}{2}(n-1)\mathrm{trace}(S\Sigma^{-1}) + \text{constant},$$

where n is the number of observations, S is the sample variance-covariance matrix or, if weights are used, S is the weighted sample variance-covariance matrix and n is the effective number of observations, that is, the sum of the weights. The constant is independent of the parameters of the model. A two stage maximization is employed. It makes use of the function $F(\Psi)$, which is, up to a constant, $-2/(n-1)$ times the log likelihood maximized over Λ. This is then minimized with respect to Ψ to give the estimates, $\hat{\Psi}$, of Ψ. The function $F(\Psi)$ can be written as:

$$F(\Psi) = \sum_{j=k+1}^{p} (\theta_j - \log\theta_j) - (p-k),$$

where values θ_j, for $j = 1, 2, \ldots, p$ are the eigenvalues of the matrix:

$$S^* = \Psi^{-1/2} S \Psi^{-1/2}.$$

The estimates $\hat{\Lambda}$, of Λ, are then given by scaling the eigenvectors of S^*, which are denoted by V:

$$\hat{\Lambda} = \Psi^{1/2} V (\Theta - I)^{1/2}.$$

where Θ is the diagonal matrix with elements θ_i, and I is the identity matrix.

The minimization of $F(\Psi)$ is performed using nag_opt_bounds_2nd_deriv (e04lbc) which uses a modified Newton algorithm. The computation of the Hessian matrix is described by Clarke (1970). However, instead of using the eigenvalue decomposition of the matrix S^* as described above, the singular value decomposition of the matrix $R\Psi^{-1/2}$ is used, where R is obtained either from the QR decomposition of the (scaled) mean-centred data matrix or from the Cholesky decomposition of the correlation/covariance matrix. The routine nag_opt_bounds_2nd_deriv (e04lbc) ensures that the values of ψ_i are greater than a given small positive quantity, δ, so that the communality is always less than one. This avoids the so called Heywood cases.

In addition to the values of Λ, Ψ and the communalities, nag_mv_factor (g03cac) returns the residual correlations, i.e., the off-diagonal elements of $C - (\Lambda\Lambda^T + \Psi)$ where C is the sample correlation matrix. nag_mv_factor (g03cac) also returns the test statistic:

$$\chi^2 = [n - 1 - (2p + 5)/6 - 2k/3] F(\hat{\Psi})$$

which can be used to test the goodness of fit of the model (1), see Lawley and Maxwell (1971) and Morrison (1967).

4. Parameters

matrix

Input: selects the type of matrix on which factor analysis is to be performed.

If **matrix** = **Nag_DataCorr** (Data input), then the data matrix will be input in **x** and factor analysis will be computed for the correlation matrix.

If **matrix** = **Nag_DataCovar**, then the data matrix will be input in **x** and factor analysis will be computed for the covariance matrix, i.e., the results are scaled as described in Section 6.

If **matrix** = **Nag_MatCorr_Covar**, then the correlation/variance-covariance matrix will be input in **x** and factor analysis computed for this matrix.

Constraint: **matrix** = **Nag_DataCorr**, **Nag_DataCovar** or **Nag_MatCorr_Covar**.

n

Input: if **matrix** = **Nag_DataCorr** or **Nag_DataCovar** the number of observations in the data array **x**.

If **matrix** = **Nag_MatCorr_Covar** the (effective) number of observations used in computing the (possibly weighted) correlation/variance-covariance matrix input in **x**.

Constraint: **n** > **nvar**.

m

Input: the number of variables in the data/correlation/variance-covariance matrix.

Constraint: **m** ≥ **nvar**.

x[*dim1*][**tdx**]

Input: the input matrix. If **matrix** = **Nag_DataCorr** or **Nag_DataCovar**, then *dim1* ≥ **n** and **x** must contain the data matrix, i.e., **x**$[i-1][j-1]$ must contain the ith observation for the jth variable, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots,$ **m**.

If **matrix** = **Nag_MatCorr_Covar** then *dim1* ≥ **m** and **x** must contain the correlation or variance-covariance matrix. Only the upper triangular part is required.

tdx

Input: the last dimension of the array **x** as declared in the calling program.
Constraint: **tdx** $\geq$ **m**.

nvar

Input: the number of variables in the factor analysis, p.
Constraint: **nvar** ≥ 2.

isx[m]

Input: **isx**$[j-1]$ indicates whether or not the jth variable is to be included in the factor analysis.

If **isx**$[j-1] \geq 1$, then the variable represented by the jth column of **x** is included in the analysis; otherwise it is excluded, for $j = 1, 2, \ldots,$ **m**.

Constraint: **isx**$[j-1] > 0$ for **nvar** values of j.

nfac

Input: the number of factors, k.
Constraint: $1 \leq$ **nfac** $\leq$ **nvar**.

wt[n]

Input: if **matrix** = **Nag_DataCorr** or **Nag_DataCovar** then the elements of **wt** must contain the weights to be used in the factor analysis. The effective number of observations is the sum of the weights. If **wt**$[i-1] = 0.0$ then the ith observation is not included in the analysis.

If **matrix** = **Nag_MatCorr_Covar** or **wt** is set to the null pointer **NULL**, i.e., (double *)0, then **wt** is not referenced and the effective number of observations is n.

Constraint: if **wt** is referenced, then **wt**$[i-1] \geq 0$ for $i = 1, 2, \ldots, n$, and the sum of the weights > **nvar**.

e[nvar]

Output: the eigenvalues θ_i, for $i = 1, 2, \ldots, p$.

stat[4]

Output: the test statistics.

stat[0] contains the value $F(\hat{\Psi})$.

stat[1] contains the test statistic, χ^2.

stat[2] contains the degrees of freedom associated with the test statistic.

stat[3] contains the significance level.

com[nvar]

Output: the communalities.

psi[nvar]

Output: the estimates of ψ_i, for $i = 1, 2, \ldots, p$.

res[nvar∗(nvar−1)/2]

Output: the residual correlations. The residual correlation for the ith and jth variables is stored in **res**$[(j-1)(j-2)/2 + i - 1]$, $i < j$.

fl[nvar][tdfl]

Output: the factor loadings. **fl**$[i-1][j-1]$ contains λ_{ij}, for $i = 1, 2, \ldots, p$; $j = 1, 2, \ldots, k$.

tdfl

Input: the last dimension of the array **fl** as declared in the calling program.
Constraint: **tdfl** $\geq$ **nfac**.

options

Input/Output: a pointer to a structure of type **Nag_E04_Opt** whose members are optional parameters for nag_opt_bounds_2nd_deriv (e04lbc). These structure members offer the means of adjusting some of the parameter values of the algorithm.

If the optional parameters are not required the NAG defined null pointer, `E04_DEFAULT`, can be used in the function call. See the document for nag_opt_bounds_2nd_deriv (e04lbc) for further details.

eps

Input: A lower bound for the value of Ψ_i.
Constraint: ***machine precision*** $\leq$ **eps** < 1.0.

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 **matrix** had an illegal value.

NE_INT_ARG_LT

On entry, **nfac** must not be less than 1: **nfac** = ⟨*value*⟩.
On entry, **nvar** must not be less than 2: **nvar** = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry, **m** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **m** $\geq$ **nvar**.
On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩.
These parameters must satisfy **tdx** $\geq$ **m**.
On entry, **tdfl** = ⟨*value*⟩ while **nfac** = ⟨*value*⟩.
These parameters must satisfy **tdfl** $\geq$ **nfac**.

NE_2_INT_ARG_LE

On entry, **n** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **n** > **nvar**.

NE_2_INT_ARG_GT

On entry, **nfac** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **nfac** $\leq$ **nvar**.

NE_INVALID_REAL_RANGE_EF

Value ⟨*value*⟩ given to **eps** is not valid.
Correct range is ***machine precision*** $\leq$ **eps** < 1.0.

NE_NEG_WEIGHT_ELEMENT

On entry, **wt**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: When referenced, all elements of **wt** must be non-negative.

NE_VAR_INCL_INDICATED

The number of variables, **nvar** in the analysis = ⟨*value*⟩, while number of variables included in the analysis via array **isx** = ⟨*value*⟩.
Constraint: these two numbers must be the same.

NE_OBSERV_LT_VAR

With weighted data, the effective number of observations given by the sum of
weights = ⟨*value*⟩, while the number of variables included in the analysis, **nvar** = ⟨*value*⟩.
Constraint: effective number of observations > **nvar** + 1.

NE_SVD_NOT_CONV

A singular value decomposition has failed to converge.
This is a very unlikely error exit.

NW_COND_MIN

The conditions for a minimum have not all been satisfied but a lower
point could not be found.
Note that in this case all the results are computed.
See nag_opt_bounds_2nd_deriv (e04lbc) for further details.

NW_TOO_MANY_ITER

The maximum number of iterations, ⟨*value*⟩, have been performed.

NE_MAT_RANK

On entry, **matrix** = **Nag_DataCorr** or **matrix** = **Nag_DataCovar** and the data matrix is not of full column rank, or **matrix** = **Nag_MatCorr_Covar** and the input correlation/variance-covariance matrix is not positive-definite.

This exit may also be caused by two of the eigenvalues of S^* being equal; this is rare (see Lawley and Maxwell (1971)) and may be due to the data/correlation matrix being almost singular.

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.

Additional error messages are output if the optimisation fails to converge or if the options are set incorrectly. Details of these can be found in the nag_opt_bounds_2nd_deriv (e04lbc) document.

6. Further Comments

The factor loadings may be orthogonally rotated by using nag_mv_orthomax (g03bac) and factor score coefficients can be computed using nag_mv_fac_score (g03ccc). The maximum likelihood estimators are invariant to a change in scale. This means that the results obtained will be the same (up to a scaling factor) if either the correlation matrix or the variance-covariance matrix is used. As the correlation matrix ensures that all values of ψ_i are between 0 and 1 it will lead to a more efficient optimization. In the situation when the data matrix is input the results are always computed for the correlation matrix and then scaled if the results for the covariance matrix are required. When the user inputs the covariance/correlation matrix the input matrix itself is used and so the user is advised to input the correlation matrix rather than the covariance matrix.

6.1. Accuracy

The accuracy achieved is discussed in nag_opt_bounds_2nd_deriv (e04lbc).

6.2. References

Clark M R B (1970) A rapidly convergent method for maximum likelihood factor analysis *British J. Math. Statist. Psych.*.

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

Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM* **20(3)** 2–25.

Morrison D F (1967) *Multivariate Statistical Methods* McGraw-Hill.

7. See Also

nag_opt_bounds_2nd_deriv (e04lbc)

8. Example

The example is taken from Lawley and Maxwell (1971). The correlation matrix for nine variables is input and the parameters of a factor analysis model with three factors are estimated and printed.

8.1. Program Text

```
/* nag_mv_factor (g03cac) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 *
 */

#include <nag.h>
```

```
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage04.h>
#include <nagg03.h>

#define NMAX 9
#define MMAX 9

main()
{
  double com[MMAX], e[MMAX], fl[MMAX][MMAX], psi[MMAX],
  res[MMAX*(MMAX-1)/2], stat[4], wt[NMAX],  x[NMAX][MMAX];
  double *wtptr=0;
  double eps;

  Integer nfac, nvar;
  Integer i, j, l, m, n;
  Integer  isx[MMAX];
  Integer tdfl = MMAX, tdx = MMAX;

  char weight[2], char_matrix[2];

  Nag_FacMat matrix;
  Nag_E04_Opt options;

  Vprintf("g03cac Example Program Results\n\n");

  /* Skip headings in data file */
  Vscanf("%*[^\n]");

  Vscanf("%s",char_matrix);
  Vscanf("%s",weight);
  Vscanf("%ld",&n);
  Vscanf("%ld",&m);
  Vscanf("%ld",&nvar);
  Vscanf("%ld",&nfac);

  if (m <= MMAX && (*char_matrix == 'C' || n <= NMAX ))
    {
      if (*char_matrix == 'C')
        {
          for (i = 0; i < m; ++i)
            {
              for (j = 0; j < m; ++j)
                Vscanf("%lf",&x[i][j]);
            }
        }
      else
        {
          if (*weight == 'W' || *weight == 'w')
            {
              for (i = 0; i < n; ++i)
                {
                  for (j = 0; j < m; ++j)
                    Vscanf("%lf",&x[i][j]);
                  Vscanf("%lf",&wt[i]);
                }
              wtptr = wt;
            }
          else
            {
              for (i = 0; i < n; ++i)
                {
                  for (j = 0; j < m; ++j)
                    Vscanf("%lf",&x[i][j]);
                }
            }
        }
      for (j = 0; j < m; ++j)
        Vscanf("%ld",&isx[j]);
```

```
          if (*char_matrix == 'D')
            {
              matrix = Nag_DataCorr;
            }
          else if (*char_matrix == 'S')
            {
              matrix = Nag_DataCovar;
            }
          else if (*char_matrix == 'C')
            {
              matrix = Nag_MatCorr_Covar;
            }

          e04xxc(&options);
          options.max_iter = 500;
          options.optim_tol = 1e-2;
          eps = 1e-5;
          g03cac(matrix, n, m, (double *)x, tdx, nvar, isx, nfac, wtptr, e,
                 stat, com, psi, res, (double *)fl, tdfl, &options, eps, NAGERR_DEFAULT);

          Vprintf("\nEigenvalues\n\n");
          for (j = 0; j < m; ++j)
            {
              Vprintf(" %12.4e%s",e[j], (j+1)%6==0 ? "\n" : "");
            }
          Vprintf("\n\n%s%6.3f\n","    Test Statistic = ",stat[1]);
          Vprintf("%s%6.3f\n","                df = ",stat[2]);
          Vprintf("%s%6.3f\n\n","Significance level = ",stat[3]);
          Vprintf("Residuals\n\n");
          l = 1;
          for (i = 1; i <= nvar-1; ++i)
            {
              for (j = l; j <= l+i-1; ++j)
                Vprintf(" %8.3f",res[j-1]);
              Vprintf("\n");
              l += i;
            }
          Vprintf("\nLoadings, Communalities and PSI\n\n");
          for (i = 0; i < nvar; ++i)
            {
              for (j = 0; j < nfac; ++j)
                Vprintf(" %8.3f",fl[i][j]);
              Vprintf("%8.3f%8.3f\n", com[i], psi[i]);
            }
          exit(EXIT_SUCCESS);
        }
      else
        {
          Vprintf("Incorrect input value of n or m.\n");
          exit(EXIT_FAILURE);
        }
    }
```

8.2. Program Data

```
g03cac Example Program Data
C U 211 9  9 3
 1.000 0.523 0.395 0.471 0.346 0.426 0.576 0.434 0.639
 0.523 1.000 0.479 0.506 0.418 0.462 0.547 0.283 0.645
 0.395 0.479 1.000 0.355 0.270 0.254 0.452 0.219 0.504
 0.471 0.506 0.355 1.000 0.691 0.791 0.443 0.285 0.505
 0.346 0.418 0.270 0.691 1.000 0.679 0.383 0.149 0.409
 0.426 0.462 0.254 0.791 0.679 1.000 0.372 0.314 0.472
 0.576 0.547 0.452 0.443 0.383 0.372 1.000 0.385 0.680
 0.434 0.283 0.219 0.285 0.149 0.314 0.385 1.000 0.470
 0.639 0.645 0.504 0.505 0.409 0.472 0.680 0.470 1.000
  1    1    1   1    1   1   1   1   1
```

8.3. Program Results

```
g03cac Example Program Results


Parameters to e04lbc
--------------------

Number of variables...........  9

optim_tol............... 1.00e-02    linesearch_tol.......... 9.00e-01
step_max................ 2.70e+01    max_iter................      500
print_level.........Nag_Soln_Iter    machine precision....... 1.11e-16
deriv_check.............    FALSE
outfile.................   stdout

Memory allocation:
state...................     User
hesl....................     User    hesd...................      User

Iterations performed = 0,   function evaluations = 1
Criterion =     8.635756e-02

             Variable    Standardized
                         Communalities
                1           0.5755
                2           0.5863
                3           0.4344
                4           0.7496
                5           0.6203
                6           0.7329
                7           0.6061
                8           0.4053
                9           0.7104

Iterations performed = 1,   function evaluations = 3
Criterion =     3.603203e-02

             Variable    Standardized
                         Communalities
                1           0.5517
                2           0.5800
                3           0.3936
                4           0.7926
                5           0.6140
                6           0.8254
                7           0.6052
                8           0.5076
                9           0.7569

Iterations performed = 2,   function evaluations = 4
Criterion =     3.502097e-02

             Variable    Standardized
                         Communalities
                1           0.5496
                2           0.5731
                3           0.3838
                4           0.7875
                5           0.6200
                6           0.8238
                7           0.6006
                8           0.5349
                9           0.7697

Iterations performed = 3,   function evaluations = 5
Criterion =     3.501729e-02

             Variable    Standardized
                         Communalities
```

```
                     1           0.5495
                     2           0.5729
                     3           0.3835
                     4           0.7877
                     5           0.6195
                     6           0.8231
                     7           0.6005
                     8           0.5384
                     9           0.7691

Eigenvalues

   1.5968e+01   4.3577e+00   1.8474e+00   1.1560e+00   1.1190e+00   1.0271e+00
   9.2574e-01   8.9508e-01   8.7710e-01

    Test Statistic =  7.149
                df = 12.000
Significance level =  0.848

Residuals

    0.000
   -0.013    0.022
    0.011   -0.005    0.023
   -0.010   -0.019   -0.016    0.003
   -0.005    0.011   -0.012   -0.001   -0.001
    0.015   -0.022   -0.011    0.002    0.029   -0.012
   -0.001   -0.011    0.013    0.005   -0.006   -0.001    0.003
   -0.006    0.010   -0.005   -0.011    0.002    0.007    0.003   -0.001

Loadings, Communalities and PSI

    0.664   -0.321    0.074   0.550   0.450
    0.689   -0.247   -0.193   0.573   0.427
    0.493   -0.302   -0.222   0.383   0.617
    0.837    0.292   -0.035   0.788   0.212
    0.705    0.315   -0.153   0.619   0.381
    0.819    0.377    0.105   0.823   0.177
    0.661   -0.396   -0.078   0.600   0.400
    0.458   -0.296    0.491   0.538   0.462
    0.766   -0.427   -0.012   0.769   0.231
```