NAG Library Function Document

nag_mv_prin_coord_analysis (g03fac)

1 Purpose

nag_mv_prin_coord_analysis (g03fac) performs a principal coordinate analysis also known as classical metric scaling.

2 Specification

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

```
void nag_mv_prin_coord_analysis (Nag_Eigenvalues roots, Integer n,
     const double d[], Integer ndim, double x[], Integer tdx, double eval[],
     NagError *fail)
```

3 Description

For a set of n objects, a distance matrix D can be calculated such that d_{ij} is a measure of how 'far apart' objects i and j are (see nag_mv_distance_mat (g03eac) for example). Principal coordinate analysis or metric scaling starts with a distance matrix and finds points X in Euclidean space such that those points have the same distance matrix. The aim is to find a small number of dimensions, $k << (n-1)$, that provide an adequate representation of the distances.

The principal coordinates of the points are computed from the eigenvectors of the matrix E where $e_{ij} = -\frac{1}{2}\left(d_{ij}^2 - d_{i.}^2 - d_{.j}^2 - d_{..}^2\right)$ with d_i^2 denoting the average of d_{ij}^2 over the suffix j etc.. The eigenvectors are then scaled by multiplying by the square root of the value of the corresponding eigenvalue.

Provided that the ordered eigenvalues, λ_i, of the matrix E are all positive, $\sum_{i=1}^{k}\lambda_i/\sum_{i=1}^{n-1}\lambda_i$ shows how well the data is represented in k dimensions. The eigenvalues will be non-negative if E is positive semidefinite. This will be true provided d_{ij} satisfies the inequality: $d_{ij} \leq d_{ik} + d_{jk}$ for all i, j, k. If this is not the case the size of the negative eigenvalue reflects the amount of deviation from this condition and the results should be treated cautiously in the presence of large negative eigenvalues. See Krzanowski (1990) for further discussion. nag_mv_prin_coord_analysis (g03fac) provides the option for all eigenvalues to be computed so that the smallest eigenvalues can be checked.

4 References

Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall

Gower J C (1966) Some distance properties of latent root and vector methods used in multivariate analysis *Biometrika* **53** 325–338

Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

5 Arguments

1: **roots** – Nag_Eigenvalues *Input*

On entry: indicates if all the eigenvalues are to be computed or just the **ndim** largest.

roots = Nag_AllEigVals
All the eigenvalues are computed.

roots = Nag_LargeEigVals
Only the largest **ndim** eigenvalues are computed.

Constraint: **roots** = Nag_AllEigVals or Nag_LargeEigVals.

2: **n** – Integer *Input*

On entry: the number of objects in the distance matrix, n.

Constraint: **n** > **ndim**.

3: **d**[**n** × (**n** − **1**)/**2**] – const double *Input*

On entry: the lower triangle of the distance matrix D stored packed by rows. That is, **d**$[(i-1)\times(i-2)/2+j-1]$ must contain d_{ij}, for $i=2,3,\dots,n$ and $j=1,2,\dots,i-1$.

Constraint: **d**$[i-1]\geq 0.0$, for $i=1,2,\dots,n(n-1)/2$.

4: **ndim** – Integer *Input*

On entry: the number of dimensions used to represent the data, k.

Constraint: **ndim** ≥ 1.

5: **x**[**n** × **tdx**] – double *Output*

Note: the (i,j)th element of the matrix X is stored in **x**$[(i-1)\times$ **tdx** $+j-1]$.

On exit: the ith row contains k coordinates for the ith point, $i=1,2,\dots,n$.

6: **tdx** – Integer *Input*

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

Constraint: **tdx** ≥ **ndim**.

7: **eval**[**n**] – double *Output*

On exit: if **roots** = Nag_AllEigVals, **eval** contains the n scaled eigenvalues of the matrix E. If **roots** = Nag_LargeEigVals, **eval** contains the largest k scaled eigenvalues of the matrix E. In both cases the eigenvalues are divided by the sum of the eigenvalues (that is, the trace of E).

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LE

On entry, **n** = ⟨*value*⟩ while **ndim** = ⟨*value*⟩. These arguments must satisfy **n** > **ndim**.

NE_2_INT_ARG_LT

On entry, **tdx** = ⟨*value*⟩ while **ndim** = ⟨*value*⟩. These arguments must satisfy **tdx** ≥ **ndim**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **roots** had an illegal value.

NE_EIGVAL

The computation of eigenvalues or eigenvectors has failed.

NE_INT_ARG_LT

On entry, **ndim** = ⟨*value*⟩.
Constraint: **ndim** ≥ 1.

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.

NE_NEG_ELEMENT

At least one element of the array **d** < 0.0.
Constraint: **d**$[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n \times (n-1)/2$.

NE_NONZERO_EIGVALS

There are fewer than **ndim** eigenvalues greater than zero. Try a smaller number of dimensions (**ndim**) or use non-metric scaling (nag_mv_ordinal_multidimscale (g03fcc)).

NE_REALARR

On entry, **d**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **d**$[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n \times (n-1)/2$.

7 Accuracy

Alternative, non-metric, methods of scaling are provided by nag_mv_ordinal_multidimscale (g03fcc).

The relationship between principal coordinates and principal components (see nag_mv_ordinal_multidimscale (g03fcc)), is discussed in Krzanowski (1990) and Gower (1966).

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

The data, given by Krzanowski (1990), are dissimilarities between water vole populations in Europe. The first two principal coordinates are computed.

10.1 Program Text

```
/* nag_mv_prin_coord_analysis (g03fac) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 * Mark 7 revised, 2001.
 * Mark 8 revised, 2004.
 *
 */

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

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

#define XTMP(I) xtmp[(I) -1]
#define YTMP(I) ytmp[(I) -1]


#define X(I, J) x[(I) *tdx + J]
int main(void)
{
  Integer         exit_status = 0, i, j, n, ndim, nn, tdx;
  char            nag_enum_arg[40];
  double          *d = 0, *e = 0, *x = 0;
  Nag_Eigenvalues roots;
  NagError        fail;

  INIT_FAIL(fail);

  printf(
          "nag_mv_prin_coord_analysis (g03fac) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  scanf("%ld", &ndim);
  scanf("%39s", nag_enum_arg);
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  roots = (Nag_Eigenvalues) nag_enum_name_to_value(nag_enum_arg);

  if (ndim >= 1 && n > ndim)
    {
      if (!(d = NAG_ALLOC(n*(n-1)/2, double)) ||
          !(e = NAG_ALLOC(n, double)) ||
          !(x = NAG_ALLOC(n*n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdx = n;
    }
  else
    {
      printf("Invalid ndim or n.\n");
      exit_status = 1;
      return exit_status;
    }
  nn = n * (n - 1) / 2;
  for (i = 0; i < nn; ++i)
    scanf("%lf", &d[i]);

  /* nag_mv_prin_coord_analysis (g03fac).
   * Principal co-ordinate analysis
   */
  nag_mv_prin_coord_analysis(roots, n, d, ndim, x, tdx, e, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_mv_prin_coord_analysis (g03fac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
  printf("\nScaled Eigenvalues\n\n");

  if (roots == Nag_LargeEigVals)
    {
      for (i = 0; i < ndim; ++i)
        printf("%10.4f", e[i]);
    }
```

```
  else
    {
      for (i = 0; i < n; ++i)
        printf("%10.4f", e[i]);
      printf("\n");
    }
  printf("\nCo-ordinates\n\n");
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < ndim; ++j)
        printf("%10.4f", X(i, j));
      printf("\n");
    }
 END:
  NAG_FREE(d);
  NAG_FREE(e);
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_mv_prin_coord_analysis (g03fac) Example Program Data

14 2 Nag_LargeEigVals

0.099
0.033 0.022
0.183 0.114 0.042
0.148 0.224 0.059 0.068
0.198 0.039 0.053 0.085 0.051
0.462 0.266 0.322 0.435 0.268 0.025
0.628 0.442 0.444 0.406 0.240 0.129 0.014
0.113 0.070 0.046 0.047 0.034 0.002 0.106 0.129
0.173 0.119 0.162 0.331 0.177 0.039 0.089 0.237 0.071
0.434 0.419 0.339 0.505 0.469 0.390 0.315 0.349 0.151 0.430
0.762 0.633 0.781 0.700 0.758 0.625 0.469 0.618 0.440 0.538 0.607
0.530 0.389 0.482 0.579 0.597 0.498 0.374 0.562 0.247 0.383 0.387 0.084
0.586 0.435 0.550 0.530 0.552 0.509 0.369 0.471 0.234 0.346 0.456 0.090 0.038
```

10.3 Program Results

```
nag_mv_prin_coord_analysis (g03fac) Example Program Results


Scaled Eigenvalues

    0.7871    0.2808
Co-ordinates

    0.2408    0.2337
    0.1137    0.1168
    0.2394    0.0760
    0.2129    0.0605
    0.2495   -0.0693
    0.1487   -0.0778
   -0.0514   -0.1623
    0.0115   -0.3446
   -0.0039    0.0059
    0.0386   -0.0089
   -0.0421   -0.0566
   -0.5158    0.0291
   -0.3180    0.1501
   -0.3238    0.0475
```