NAG Library Function Document

nag_mv_procustes (g03bcc)

1 Purpose

nag_mv_procustes (g03bcc) computes Procrustes rotations in which an orthogonal rotation is found so that a transformed matrix best matches a target matrix.

2 Specification

```
#include <nag.h>
#include <nagg03.h>
void nag_mv_procustes (Nag_TransNorm stand, Nag_RotationScale pscale,
     Integer n, Integer m, double x[], Integer tdx, double y[], Integer tdy,
     double yhat[], double r[], Integer tdr, double *alpha, double *rss,
     double res[], NagError *fail)
```

3 Description

Let X and Y be n by m matrices. They can be considered as representing sets of n points in an m-dimensional space. The X matrix may be a matrix of loadings from say factor or canonical variate analysis, and the Y matrix may be a postulated pattern matrix or the loadings from a different sample. The problem is to relate the two sets of points without disturbing the relationships between the points in each set. This can be achieved by translating, rotating and scaling the sets of points. The Y matrix is considered as the target matrix and the X matrix is rotated to match that matrix.

First the two sets of points are translated so that their centroids are at the origin to give X_c and Y_c, i.e., the matrices will have zero column means. Then the rotation of the translated X_c matrix which minimizes the sum of squared distances between corresponding points in the two sets is found. This is computed from the singular value decomposition of the matrix:

$$X_c^{\mathrm{T}} Y_c = UDV^{\mathrm{T}},$$

where U and V are orthogonal matrices and D is a diagonal matrix. The matrix of rotations, R, is computed as:

$$R = UV^{\mathrm{T}}.$$

After rotation, a scaling or dilation factor, α, may be estimated by least squares. Thus, the final set of points that best match Y_c is given by:

$$\hat{Y_c} = \alpha X_c R.$$

Before rotation, both sets of points may be normalized to have unit sums of squares or the X matrix may be normalized to have the same sum of squares as the Y matrix. After rotation, the results may be translated to the original Y centroid.

The ith residual, r_i, is given by the distance between the point given in the ith row of Y and the point given in the ith row of $\hat{Y}$. The residual sum of squares is also computed.

4 References

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

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

5 Arguments

1: **stand** – Nag_TransNorm *Input*

On entry: indicates if translation/normalization is required.

stand = Nag_NoTransNorm
No translation or normalization.

stand = Nag_Orig
Translation to the origin.

stand = Nag_OrigCentroid
Translation to the origin and then to the Y centroid after rotation.

stand = Nag_Norm
Unit normalization.

stand = Nag_OrigNorm
Translation and normalization.

stand = Nag_OrigNormCentroid
Translation and normalization to Y scale, then translation to the Y centroid after rotation.

Constraint: **stand** = Nag_NoTransNorm, Nag_Orig, Nag_OrigCentroid, Nag_Norm, Nag_OrigNorm or Nag_OrigNormCentroid.

2: **pscale** – Nag_RotationScale *Input*

On entry: indicates if least squares scaling is applied after rotation.

pscale = Nag_LsqScale
Scaling is to be applied.

pscale = Nag_NotLsqScale
No scaling is applied.

Constraint: **pscale** = Nag_LsqScale or Nag_NotLsqScale.

3: **n** – Integer *Input*

On entry: the number of points, n.

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

4: **m** – Integer *Input*

On entry: the number of dimensions, m.

Constraints:

$\mathbf{m} \geq 1$;
$\mathbf{m} \leq \mathbf{n}$.

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

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

On entry: the matrix to be rotated, X.

On exit: if **stand** = Nag_NoTransNorm, **x** will be unchanged.

If **stand** = Nag_Orig, Nag_OrigCentroid, Nag_OrigNorm or Nag_OrigNormCentroid, **x** will be translated to have zero column means.

If **stand** = Nag_Norm or Nag_OrigNorm, **x** will be scaled to have unit sum of squares.

If **stand** = Nag_OrigNormCentroid, **x** will be scaled to have the same sum of squares as **y**.

6: **tdx** – Integer *Input*

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

Constraint: **tdx** $\geq$ **m**.

7: **y**[**n** $\times$ **tdy**] – double *Input/Output*

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

On entry: the target matrix, Y.

On exit: if **stand** = Nag_NoTransNorm, then **y** will be unchanged.

If **stand** = Nag_Orig or Nag_OrigNorm, then **y** will be translated to have zero column means.

If **stand** = Nag_Norm or Nag_OrigNorm, then **y** will be scaled to have unit sum of squares.

If **stand** = Nag_OrigCentroid or Nag_OrigNormCentroid, then **y** will be translated and then after rotation, translated back. The output **y** should be the same as the input **y** except for rounding errors.

8: **tdy** – Integer *Input*

On entry: the stride separating matrix column elements in the arrays **y**, **yhat**.

Constraint: **tdy** $\geq$ **m**.

9: **yhat**[**n** $\times$ **tdy**] – double *Output*

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

On exit: the fitted matrix, $\hat{Y}$.

10: **r**[**m** $\times$ **tdr**] – double *Output*

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

On exit: the matrix of rotations, R, see Section 9.

11: **tdr** – Integer *Input*

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

Constraint: **tdr** $\geq$ **m**.

12: **alpha** – double * *Output*

On exit: if **pscale** = Nag_LsqScale the scaling factor, α; otherwise **alpha** is not set.

13: **rss** – double * *Output*

On exit: the residual sum of squares.

14: **res**[**n**] – double *Output*

On exit: the residuals, r_i, for $i = 1, 2, \ldots, n$.

15: **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_GT

On entry, **m** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **m** $\leq$ **n**.

NE_2_INT_ARG_LT

On entry, **tdr** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdr** $\geq$ **m**.

On entry, **tdx** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdx** $\geq$ **m**.

On entry, **tdy** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdy** $\geq$ **m**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

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

NE_INT_ARG_LT

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

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

The fitted matrix $\hat{Y}$, contains only zero-points when least squares scaling is applied.

NE_NORM_ZERO_PTS

On entry, either **x** or **y** contains only zero-points (possibly after translation) when normalization is to be applied.

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge. This is an unlikely error exit.

7 Accuracy

The accuracy of the calculation of the rotation matrix largely depends upon the singular value decomposition. See the f08 Chapter Introduction for further details.

8 Parallelism and Performance

Not applicable.

9 Further Comments

Note that if the matrix $X_c^{\mathrm{T}}Y$ is not of full rank, then the matrix of rotations, R, may not be unique even if there is a unique solution in terms of the rotated matrix, $\hat{Y}_c$. The matrix R may also include reflections as well as pure rotations, see Krzanowski (1990).

If the column dimensions of the X and Y matrices are not equal, the smaller of the two should be supplemented by columns of zeros. Adding a column of zeros to both X and Y will have the effect of allowing reflections as well as rotations.

10 Example

Three points representing the vertices of a triangle in two dimensions are input. The points are translated and rotated to match the triangle given by (0,0),(1,0),(0,2) and scaling is applied after rotation. The target matrix and fitted matrix are printed along with additional information.

10.1 Program Text

```
/* nag_mv_procustes (g03bcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 * Mark 8 revised, 2004.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define R(I, J)    r[(I) *tdr + J]
#define X(I, J)    x[(I) *tdx + J]
#define Y(I, J)    y[(I) *tdy + J]
#define YHAT(I, J) yhat[(I) *tdy + J]
int main(void)
{
  Integer           exit_status = 0, i, j, m, n, tdr, tdx, tdy;
  char              nag_enum_arg[40];
  double            alpha, *r = 0, *res = 0, rss, *x = 0, *y = 0, *yhat = 0;
  Nag_RotationScale scale;
  Nag_TransNorm     stand;
  NagError          fail;

  INIT_FAIL(fail);

  printf("nag_mv_procustes (g03bcc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &n);
#else
  scanf("%"NAG_IFMT"", &n);
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &m);
#else
  scanf("%"NAG_IFMT"", &m);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  stand = (Nag_TransNorm) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
```

```
  scale = (Nag_RotationScale) nag_enum_name_to_value(nag_enum_arg);
  if (m >= 1 && n >= m)
    {
      if (!(r = NAG_ALLOC(m*m, double)) ||
          !(res = NAG_ALLOC(n, double)) ||
          !(x = NAG_ALLOC(n*m, double)) ||
          !(y = NAG_ALLOC(n*m, double)) ||
          !(yhat = NAG_ALLOC(n*m, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdr = m;
      tdx = m;
      tdy = m;
    }
  else
    {
      printf("Invalid m or n.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%lf", &X(i, j));
#else
        scanf("%lf", &X(i, j));
#endif
    }
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%lf", &Y(i, j));
#else
        scanf("%lf", &Y(i, j));
#endif
    }

  /* nag_mv_procustes (g03bcc).
   * Procrustes rotations
   */
  nag_mv_procustes(stand, scale, n, m, x, tdx, y, tdy,
                   yhat, r, tdr, &alpha, &rss, res, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_mv_procustes (g03bcc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n          Rotation Matrix\n\n");
  for (i = 0; i < m; ++i)
    {
      for (j = 0; j < m; ++j)
        printf("  %7.3f  ", R(i, j));
      printf("\n");
    }
  if (scale == Nag_LsqScale)
    {
      printf("\n%s%10.3f\n", " Scale factor = ", alpha);
    }
  printf("\n         Target Matrix \n\n");
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < m; ++j)
        printf("  %7.3f  ", Y(i, j));
```

```
      printf("\n");
    }
  printf("\n         Fitted Matrix\n\n");
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < m; ++j)
        printf("  %7.3f  ", YHAT(i, j));
      printf("\n");
    }
  printf("\n%s%10.3f\n", "RSS = ", rss);
 END:
  NAG_FREE(r);
  NAG_FREE(res);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(yhat);
  return exit_status;
}
```

10.2 Program Data

```
nag_mv_procustes (g03bcc) Example Program Data
 3 2 Nag_OrigCentroid Nag_LsqScale
 0.63 0.58
 1.36 0.39
 1.01 1.76
 0.0 0.0
 1.0 0.0
 0.0 2.0
```

10.3 Program Results

```
nag_mv_procustes (g03bcc) Example Program Results


          Rotation Matrix

    0.967      0.254
   -0.254      0.967

 Scale factor =      1.556

         Target Matrix

    0.000      0.000
    1.000      0.000
    0.000      2.000

         Fitted Matrix

   -0.093      0.024
    1.080      0.026
    0.013      1.950

RSS =      0.019
```