nag_mv_kmeans_cluster_analysis (g03efc)

1. Purpose

nag_mv_kmeans_cluster_analysis (g03efc) performs K-means cluster analysis.

2. Specification

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

void nag_mv_kmeans_cluster_analysis(Integer n, Integer m,  double x[],
            Integer tdx, Integer isx[], Integer nvar, Integer k,
            double cmeans[], Integer tdc, double wt[],
            Integer inc[], Integer nic[], double css[],
            double csw[], Integer maxit, NagError *fail)
```

3. Description

Given n objects with p variables measured on each object, x_{ij} for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, p$, nag_mv_kmeans_cluster_analysis allocates each object to one of K groups or clusters to minimize the within-cluster sum of squares:

$$\sum_{k=1}^{K} \sum_{i \in S_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2,$$

where S_k is the set of objects in the kth cluster and $\bar{x}_{kj}$ is the mean for the variable j over cluster k. This is often known as K-means clustering.

In addition to the data matrix, a K by p matrix giving the initial cluster centres for the K clusters is required. The objects are then initially allocated to the cluster with the nearest cluster mean. Given the initial allocation, the procedure is to iteratively search for the K-partition with locally optimal within-cluster sum of squares by moving points from one cluster to another.

Optionally, weights for each object, w_i, can be used so that the clustering is based on within-cluster weighted sums of squares:

$$\sum_{k=1}^{K} \sum_{i \in S_k} \sum_{j=1}^{p} w_i (x_{ij} - \tilde{x}_{kj})^2,$$

where $\tilde{x}_{kj}$ is the weighted mean for variable j over cluster k.

The routine is based on the algorithm of Hartigan and Wong (1979).

4. Parameters

n

Input: the number of observations, n.

Constraint: **n** ≥ 2.

m

Input: the number of variables in the array **x**.

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

x[n][tdx]

Input: **x**$[i-1][j-1]$ must contain the value of jth variable for the ith object for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots,$**m**.

tdx

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

isx[m]
Input: **isx**$[j-1]$ indicates whether or not the jth variable is to be included in the analysis.
If **isx**$[j-1] > 0$, then the jth variable contained in the jth column of **x** is included, for $j = 1, 2, \ldots,$**m**.
Constraint: **isx**$[j-1] > 0$ for **nvar** values of j.

nvar
Input: the number of variables included in the sum of squares calculations, p.
Constraint: $1 \leq$ **nvar** $\leq$ **m**.

k
Input: the number of clusters, K.
Constraint: **k** $\geq$2.

cmeans[k][tdc]
Input: **cmeans**$[i-1][j-1]$ must contain the value of the jth variable for the ith initial cluster centre, for $i = 1, 2, \ldots, K$; $j = 1, 2, \ldots, p$.
Output: **cmeans**$[i-1][j-1]$ contains the value of the jth variable for the ith computed cluster centre, for $i = 1, 2, \ldots, K$; $j = 1, 2, \ldots, p$.

tdc
Input: the last dimension of the array **cmeans** as declared in the calling program.
Constraint: **tdc** $\geq$ **nvar**.

wt[n]
Input: the elements of **wt** must contain the weights to be used in the 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.
Constraint: **wt**$[i-1] \geq 0.0$ for $i = 1, 2, \ldots, n$ and **wt**$[i-1] > 0.0$ for at least two values of i.
Note: if **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.

inc[n]
Output: **inc**$[i-1]$ contains the cluster to which the ith object has been allocated, for $i = 1, 2, \ldots, n$.

nic[k]
Output: **nic**$[i-1]$ contains the number of objects in the ith cluster, for $i = 1, 2, \ldots, K$.

css[k]
Output: **css**$[i-1]$ contains the within-cluster (weighted) sum of squares of the ith cluster, for $i = 1, 2, \ldots, K$.

csw[k]
Output: **csw**$[i-1]$ contains the within-cluster sum of weights of the ith cluster, for $i = 1, 2, \ldots, K$. If **wt** = **NULL** the sum of weights is the number of objects in the cluster.

maxit
Input: the maximum number of iterations allowed in the analysis.
Constraint: **maxit** > 0.
Suggested Value: **maxit** $= 10$.

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

5. Error Indications and Warnings

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

NE_INT_ARG_LE
On entry, **maxit** must not be less than or equal to 0: **maxit** = ⟨*value*⟩.

NE_2_INT_ARG_LT
On entry, **m** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **m** ≥ **nvar**.
On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩.
These parameters must satisfy **tdx** ≥ **m**.
On entry, **tdc** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **tdc** ≥ **nvar**.

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

NE_WT_ZERO
At least two elements of **wt** must be greater than zero.

NE_CLUSTER_EMPTY
At least one cluster is empty after the initial assignment.

Try a different set of initial cluster centres in **cmeans** and also consider decreasing the value of **k**. The empty clusters may be found by examining the values in **nic**.

NE_TOO_MANY
Too many iterations (⟨*value*⟩).
Convergence has not been achieved within the maximum number of iterations given by **maxit**. Try increasing **maxit** and, if possible, use the returned values in **cmeans** as the initial cluster centres.

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

The time per iteration is approximately proportional to npK.

6.1. Accuracy

The routine produces clusters that are locally optimal; the within-cluster sum of squares may not be decreased by transfering a point from one cluster to another, but different partitions may have the same or smaller within-cluster sum of squares.

6.2. References

Everitt B S (1974) *Cluster Analysis* Heinemann.
Hartigan J A and Wong M A (1979) Algorithm AS136: A K-means clustering algorithm *Appl. Statist.* **28** 100–108.
Kendall M G and Stuart A (1976) *The Advanced Theory of Statistics (Volume 3)* Griffin (3rd Edition).
Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press.

7. See Also

None.

8. Example

The data consists of observations of five variables on twenty soils (Kendall and Stuart (1976)). The data is read in, the K-means clustering performed and the results printed.

8.1. Program Text

```
/* nag_mv_kmeans_cluster_analysis (g03efc) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 *
 */

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

#define NMAX 20
#define MMAX 5
#define KMAX 3

main()
{

  double cmeans[KMAX][MMAX], css[MMAX], csw[MMAX],
  wt[NMAX], x[NMAX][MMAX];
  double *wtptr;

  Integer nvar, i, j, k;
  Integer m, n;
  Integer inc[NMAX], isx[MMAX], nic[MMAX];
  Integer maxit;
  Integer tdc=MMAX, tdx=MMAX;

  char weight[2];

  Vprintf("g03efc Example Program Results\n\n");

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

  Vscanf("%s",weight);
  Vscanf("%ld",&n);
  Vscanf("%ld",&m);
  Vscanf("%ld",&nvar);
  Vscanf("%ld",&k);
  Vscanf("%ld",&maxit);

  if (n <= NMAX && m <= MMAX)
    {
      if (*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]);
            }
          wtptr = 0;
```

```
        }
      for (i = 0; i < k; ++i)
        {
          for (j = 0; j < nvar; ++j)
            Vscanf("%lf",&cmeans[i][j]);
        }
      for (j = 0; j < m; ++j)
        Vscanf ("%ld",&isx[j]);

      g03efc(n, m, (double *)x, tdx, isx, nvar, k, (double *)cmeans,
             tdc, wtptr, inc, nic, css, csw, maxit, NAGERR_DEFAULT);

      Vprintf("\nThe cluster each point belongs to\n");
      for (i = 0; i < n; ++i)
        Vprintf(" %6ld%s",inc[i], (i+1)%10 ? "" : "\n");

      Vprintf("\n\nThe number of points in each cluster\n");
      for (i = 0; i < k; ++i)
        Vprintf(" %6ld",nic[i]);

      Vprintf("\n\nThe within-cluster sum of weights of each cluster\n");
      for (i = 0; i < k; ++i)
        Vprintf(" %9.2f",csw[i]);

      Vprintf("\n\nThe within-cluster sum of squares of each cluster\n\n");
      for (i = 0; i < k; ++i)
        Vprintf(" %13.4f",css[i]);

      Vprintf("\n\nThe final cluster centres\n");
      Vprintf("                    1       2       3       4       5\n");
      for (i = 0; i < k; ++i)
        {
          Vprintf(" %5ld     ",i+1);
          for (j = 0; j < nvar; ++j)
            Vprintf("%8.4f",cmeans[i][j]);
          printf("\n");
        }
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vprintf("Incorrect input value of n or m.\n");
      exit(EXIT_FAILURE);
    }
}
```

8.2. Program Data

```
g03efc Example Program Data

U 20 5 5 3 10

77.3 13.0  9.7 1.5 6.4
82.5 10.0  7.5 1.5 6.5
66.9 20.6 12.5 2.3 7.0
47.2 33.8 19.0 2.8 5.8
65.3 20.5 14.2 1.9 6.9
83.3 10.0  6.7 2.2 7.0
81.6 12.7  5.7 2.9 6.7
47.8 36.5 15.7 2.3 7.2
48.6 37.1 14.3 2.1 7.2
61.6 25.5 12.9 1.9 7.3
58.6 26.5 14.9 2.4 6.7
69.3 22.3  8.4 4.0 7.0
61.8 30.8  7.4 2.7 6.4
67.7 25.3  7.0 4.8 7.3
57.2 31.2 11.6 2.4 6.5
67.2 22.7 10.1 3.3 6.2
59.2 31.2  9.6 2.4 6.0
80.2 13.2  6.6 2.0 5.8
```

```
82.2 11.1  6.7 2.2 7.2
69.7 20.7  9.6 3.1 5.9

82.5 10.0  7.5 1.5 6.5
47.8 36.5 15.7 2.3 7.2
67.2 22.7 10.1 3.3 6.2

1 1 1 1 1
```

8.3. Program Results

```
g03efc Example Program Results


The cluster each point belongs to
     1      1      3      2      3      1      1      2      2      3
     3      3      3      3      3      3      3      1      1      3


The number of points in each cluster
     6      3     11

The within-cluster sum of weights of each cluster
     6.00      3.00     11.00

The within-cluster sum of squares of each cluster

      46.5717       20.3800      468.8964

The final cluster centres
                 1       2       3       4       5
     1      81.1833 11.6667  7.1500  2.0500  6.6000
     2      47.8667 35.8000 16.3333  2.4000  6.7333
     3      64.0455 25.2091 10.7455  2.8364  6.6545
```