

NAG Library Function Document

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, const double x[],
    Integer tdx, const Integer isx[], Integer nvar, Integer k,
    double cmeans[], Integer tdc, const 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 $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, p$, nag_mv_kmeans_cluster_analysis (g03efc) 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 k th 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 function is based on the algorithm of Hartigan and Wong (1979).

4 References

Everitt B S (1974) *Cluster Analysis* Heinemann

Hartigan J A and Wong M A (1979) Algorithm AS 136: A K -means clustering algorithm *Appl. Statist.* **28** 100–108

Kendall M G and Stuart A (1976) *The Advanced Theory of Statistics (Volume 3)* (3rd Edition) Griffin

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

5 Arguments

- 1: **n** – Integer *Input*
On entry: the number of observations, n .
Constraint: $n \geq 2$.
- 2: **m** – Integer *Input*
On entry: the number of variables in the array **x**.
Constraint: $m \geq \mathbf{nvar}$.
- 3: **x**[$n \times \mathbf{tdx}$] – const double *Input*
On entry: **x**[($i - 1$) \times \mathbf{tdx} + $j - 1$] must contain the value of j th variable for the i th object, for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.
- 4: **tdx** – Integer *Input*
On entry: the stride separating matrix column elements in the array **x**.
Constraint: $\mathbf{tdx} \geq m$.
- 5: **isx**[**m**] – const Integer *Input*
On entry: **isx**[$j - 1$] indicates whether or not the j th variable is to be included in the analysis.
 If **isx**[$j - 1$] > 0 , then the j th variable contained in the j th column of **x** is included, for $j = 1, 2, \dots, m$.
Constraint: **isx**[$j - 1$] > 0 for **nvar** values of j .
- 6: **nvar** – Integer *Input*
On entry: the number of variables included in the sum of squares calculations, p .
Constraint: $1 \leq \mathbf{nvar} \leq m$.
- 7: **k** – Integer *Input*
On entry: the number of clusters, K .
Constraint: $k \geq 2$.
- 8: **cmeans**[$k \times \mathbf{tdc}$] – double *Input/Output*
On entry: **cmeans**[($i - 1$) \times \mathbf{tdc} + $j - 1$] must contain the value of the j th variable for the i th initial cluster centre, for $i = 1, 2, \dots, K$ and $j = 1, 2, \dots, p$.
On exit: **cmeans**[($i - 1$) \times \mathbf{tdc} + $j - 1$] contains the value of the j th variable for the i th computed cluster centre, for $i = 1, 2, \dots, K$ and $j = 1, 2, \dots, p$.
- 9: **tdc** – Integer *Input*
On entry: the stride separating matrix column elements in the array **cmeans**.
Constraint: $\mathbf{tdc} \geq \mathbf{nvar}$.
- 10: **wt**[**n**] – const double *Input*
On entry: 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 i th observation is not included in the analysis.
 If weights are not provided then **wt** must be set to **NULL** and the effective number of observations is **n**.

Constraints:

$$\begin{aligned} \mathbf{wt}[i-1] &\geq 0.0, \text{ for } i = 1, 2, \dots, n; \\ \mathbf{wt}[i-1] &> 0.0 \text{ for at least two values of } i. \end{aligned}$$

- 11: **inc[n]** – Integer *Output*
On exit: **inc**[*i* – 1] contains the cluster to which the *i*th object has been allocated, for *i* = 1, 2, ..., *n*.
- 12: **nic[k]** – Integer *Output*
On exit: **nic**[*i* – 1] contains the number of objects in the *i*th cluster, for *i* = 1, 2, ..., *K*.
- 13: **css[k]** – double *Output*
On exit: **css**[*i* – 1] contains the within-cluster (weighted) sum of squares of the *i*th cluster, for *i* = 1, 2, ..., *K*.
- 14: **csw[k]** – double *Output*
On exit: **csw**[*i* – 1] contains the within-cluster sum of weights of the *i*th cluster, for *i* = 1, 2, ..., *K*. If **wt** = **NULL** the sum of weights is the number of objects in the cluster.
- 15: **maxit** – Integer *Input*
On entry: the maximum number of iterations allowed in the analysis.
Suggested value: **maxit** = 10.
Constraint: **maxit** > 0.
- 16: **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_LT

On entry, **m** = *<value>* while **nvar** = *<value>*. These arguments must satisfy **m** ≥ **nvar**.

On entry, **tdc** = *<value>* while **nvar** = *<value>*. These arguments must satisfy **tdc** ≥ **nvar**.

On entry, **tdx** = *<value>* while **m** = *<value>*. These arguments must satisfy **tdx** ≥ **m**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

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_INT_ARG_LE

On entry, **maxit** = *<value>*.

Constraint: **maxit** > 0.

NE_INT_ARG_LT

On entry, **k** = *<value>*.

Constraint: **k** ≥ 2.

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 2 .

On entry, **nvar** = $\langle value \rangle$.

Constraint: **nvar** ≥ 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_WEIGHT_ELEMENT

On entry, **wt**[$\langle value \rangle$] = $\langle value \rangle$.

Constraint: When referenced, all elements of **wt** must be non-negative.

NE_TOO_MANY

Too many iterations ($\langle value \rangle$). 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_VAR_INCL_INDICATED

The number of variables, **nvar** in the analysis = $\langle value \rangle$, while number of variables included in the analysis via array **isx** = $\langle value \rangle$.

Constraint: these two numbers must be the same.

NE_WT_ZERO

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

7 Accuracy

`nag_mv_kmeans_cluster_analysis` (g03efc) produces clusters that are locally optimal; the within-cluster sum of squares may not be decreased by transferring a point from one cluster to another, but different partitions may have the same or smaller within-cluster sum of squares.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time per iteration is approximately proportional to npK .

10 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.

10.1 Program Text

```
/* nag_mv_kmeans_cluster_analysis (g03efc) 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 CMEANS(I, J) cmeans[(I) *tdcmeans + J]
#define X(I, J) x[(I) *tdx + J]
int main(void)
{
    Integer    exit_status = 0, i, *inc = 0, *isx = 0, j, k, m, maxit, n, *nic = 0,
              nvar;
    Integer    tdcmeans, tdx;
    NagError   fail;
    char       weight[2];
    double     *cmeans = 0, *css = 0, *csw = 0, *wt = 0, *wtptr, *x = 0;

    INIT_FAIL(fail);

    printf(
        "nag_mv_kmeans_cluster_analysis (g03efc) Example Program Results"
        "\n\n");

    /* Skip heading in the data file */
    scanf("%*[^\\n]");
    scanf("%ls", weight);
    scanf("%ld", &n);
    scanf("%ld", &m);
    scanf("%ld", &nvar);
    scanf("%ld", &k);
    scanf("%ld", &maxit);

    if (n >= 2 && nvar >= 1 && m >= nvar && k >= 2)
    {
        if (!(cmeans = NAG_ALLOC((k)*(nvar), double)) ||
            !(css = NAG_ALLOC(k, double)) ||
            !(csw = NAG_ALLOC(k, double)) ||
            !(wt = NAG_ALLOC(n, double)) ||
            !(x = NAG_ALLOC((n)*(m), double)) ||
            !(inc = NAG_ALLOC(n, Integer)) ||
            !(isx = NAG_ALLOC(m, Integer)) ||
            !(nic = NAG_ALLOC(k, Integer)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
        tdx = m;
        tdcmeans = nvar;
    }
    else
    {
        printf("Invalid n or nvar or m or k.\n");
        exit_status = 1;
        return exit_status;
    }
    if (*weight == 'W')
    {
        for (i = 0; i < n; ++i)
        {
            for (j = 0; j < m; ++j)
                scanf("%lf", &X(i, j));
            scanf("%lf", &wt[i]);
        }
        wtptr = wt;
    }
    else
    {
        for (i = 0; i < n; ++i)
        {
            for (j = 0; j < m; ++j)

```

```

        scanf("%lf", &X(i, j));
    }
    wtptr = 0;
}
for (i = 0; i < k; ++i)
{
    for (j = 0; j < nvar; ++j)
        scanf("%lf", &CMEANS(i, j));
}
for (j = 0; j < m; ++j)
    scanf("%ld", &isx[j]);

/* nag_mv_kmeans_cluster_analysis (g03efc).
 * K-means
 */
nag_mv_kmeans_cluster_analysis(n, m, x, tdx, isx, nvar, k, cmeans,
                               tdcmeans, wtptr, inc, nic, css, csw, maxit,
                               &fail);

if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_mv_kmeans_cluster_analysis (g03efc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

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

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

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

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

printf("\n\nThe final cluster centres\n");
printf("          1          2          3          4          5\n");
for (i = 0; i < k; ++i)
{
    printf(" %5ld          ", i+1);
    for (j = 0; j < nvar; ++j)
        printf("%8.4f", CMEANS(i, j));
    printf("\n");
}
}
END:
NAG_FREE(cmeans);
NAG_FREE(css);
NAG_FREE(csw);
NAG_FREE(wt);
NAG_FREE(x);
NAG_FREE(inc);
NAG_FREE(isx);
NAG_FREE(nic);
return exit_status;
}

```

10.2 Program Data

nag_mv_kmeans_cluster_analysis (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

```

10.3 Program Results

nag_mv_kmeans_cluster_analysis (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

```
