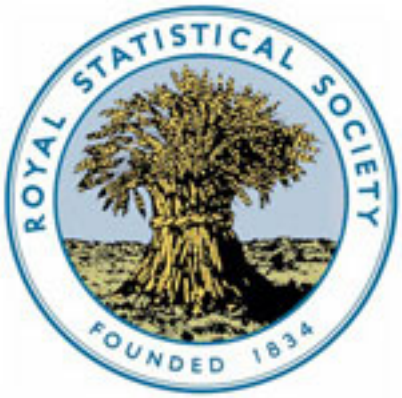

Algorithm AS 136: A K-Means Clustering Algorithm
Author(s): J. A. Hartigan and M. A. Wong
Reviewed work(s):
Source: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, No. 1 (1979), pp. 100-108
Published by: Wiley-Blackwell for the Royal Statistical Society
Stable URL: http://www.jstor.org/stable/2346830
Accessed: 22/10/2012 06:29

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at http://www.jstor.org/page/info/about/policies/terms.jsp

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.

Wiley-Blackwell and *Royal Statistical Society* are collaborating with JSTOR to digitize, preserve and extend access to *Journal of the Royal Statistical Society. Series C (Applied Statistics)*.

http://www.jstor.org

```
C        FIND MAXIMUM ENTRY
C
   60 PIVOT = ACU
      KK = 0
      DO 70 I = II, M
      K = INDEX(I)
      IF (ABS(LU(K, II)) .LE. PIVOT) GOTO 70
      PIVOT = ABS(LU(K, II))
      KK = I
   70 CONTINUE
      IF (KK .EQ. 0) GOTO 10
C
C        SWITCH ORDER
C
      ISAVE = INDEX(KK)
      INDEX(KK) = INDEX(II)
      INDEX(II) = ISAVE
C
C        PUT IN COLUMNS OF LU ONE AT A TIME
C
      IF (INTL) IBASE(II) = IROW
      IF (II .EQ. M) GOTO 90
      J = II + 1
      DO 80 I = J, M
      K = INDEX(I)
      LU(K, II) = LU(K, II) / LU(ISAVE, II)
   80 CONTINUE
   90 CONTINUE
      KKK = IROW
      RETURN
      END
```

Algorithm AS 136

A *K*-Means Clustering Algorithm

By J. A. HARTIGAN and M. A. WONG

Yale University, New Haven, Connecticut, U.S.A.

Keywords: *K*-MEANS CLUSTERING ALGORITHM; TRANSFER ALGORITHM

LANGUAGE

ISO Fortran

DESCRIPTION AND PURPOSE

The *K*-means clustering algorithm is described in detail by Hartigan (1975). An efficient version of the algorithm is presented here.

The aim of the *K*-means algorithm is to divide M points in N dimensions into K clusters so that the within-cluster sum of squares is minimized. It is not practical to require that the solution has minimal sum of squares against all partitions, except when M, N are small and $K = 2$. We seek instead "local" optima, solutions such that no movement of a point from one cluster to another will reduce the within-cluster sum of squares.

METHOD

The algorithm requires as input a matrix of M points in N dimensions and a matrix of K initial cluster centres in N dimensions. The number of points in cluster L is denoted by $NC(L)$. $D(I, L)$ is the Euclidean distance between point I and cluster L. The general procedure is to search for a K-partition with locally optimal within-cluster sum of squares by moving points from one cluster to another.

Step 1. For each point I ($I = 1, 2, \ldots, M$), find its closest and second closest cluster centres, $IC1(I)$ and $IC2(I)$ respectively. Assign point I to cluster $IC1(I)$.

Step 2. Update the cluster centres to be the averages of points contained within them.

Step 3. Initially, all clusters belong to the live set.

Step 4. This is the optimal-transfer (*OPTRA*) stage:
Consider each point I ($I = 1, 2, \ldots, M$) in turn. If cluster L ($L = 1, 2, \ldots, K$) is updated in the last quick-transfer (*QTRAN*) stage, then it belongs to the live set throughout this stage. Otherwise, at each step, it is not in the live set if it has not been updated in the last M optimal-transfer steps. Let point I be in cluster $L1$. If $L1$ is in the live set, do *Step* 4a; otherwise, do *Step* 4b.

Step 4a. Compute the minimum of the quantity, $R2 = [NC(L) * D(I,L)^2]/[NC(L)+1]$, over all clusters L ($L \neq L1, L = 1, 2, \ldots, K$). Let $L2$ be the cluster with the smallest $R2$. If this value is greater than or equal to $[NC(L1) * D(I,L1)^2]/[NC(L1)-1]$, no reallocation is necessary and $L2$ is the new $IC2(I)$. (Note that the value $[NC(L1) * D(I,L1)^2]/[NC(L1)-1]$ is remembered and will remain the same for point I until cluster $L1$ is updated.) Otherwise, point I is allocated to cluster $L2$ and $L1$ is the new $IC2(I)$. Cluster centres are updated to be the means of points assigned to them if reallocation has taken place. The two clusters that are involved in the transfer of point I at this particular step are now in the live set.

Step 4b. This step is the same as *Step* 4a, except that the minimum $R2$ is computed only over clusters in the live set.

Step 5. Stop if the live set is empty. Otherwise, go to *Step* 6 after one pass through the data set.

Step 6. This is the *quick-transfer* (*QTRAN*) stage:
Consider each point I ($I = 1, 2, \ldots, M$) in turn. Let $L1 = IC1(I)$ and $L2 = IC2(I)$. It is not necessary to check the point I if both the clusters $L1$ and $L2$ have not changed in the last M steps. Compute the values

$$R1 = [NC(L1) * D(I,L1)^2]/[NC(\mathrm{L1})-1] \quad \text{and} \quad R2 = [NC(L2) * D(I,L2)^2]/[NC(L2)+1].$$

(As noted earlier, $R1$ is remembered and will remain the same until cluster $L1$ is updated.) If $R1$ is less than $R2$, point I remains in cluster $L1$. Otherwise, switch $IC1(I)$ and $IC2(I)$ and update the centres of clusters $L1$ and $L2$. The two clusters are also noted for their involvement in a transfer at this step.

Step 7. If no transfer took place in the last M steps, go to *Step* 4. Otherwise, go to *Step* 6.

STRUCTURE

SUBROUTINE KMNS (A, M, N, C, K, IC1, IC2, NC, AN1, AN2, NCP, D, ITRAN, LIVE, ITER, WSS, IFAULT)

Formal parameters

A	Real array (*M*, *N*)	input:	the data matrix
M	Integer	input:	the number of points
N	Integer	input:	the number of dimensions
C	Real array (*K*, *N*)	input:	the matrix of initial cluster centres
		output:	the matrix of final cluster centres
K	Integer	input:	the number of clusters
IC1	Integer array (*M*)	output:	the cluster each point belongs to
IC2	Integer array (*M*)	workspace:	this array is used to remember the cluster which each point is most likely to be transferred to at each step
NC	Integer array (*K*)	output:	the number of points in each cluster
AN1	Real array (*K*)	workspace:	
AN2	Real array (*K*)	workspace:	

NCP	Integer array (K)	workspace:	
D	Real array (M)	workspace:	
ITRAN	Integer array (K)	workspace:	
LIVE	Integer array (K)	workspace:	
ITER	Integer	input:	the maximum number of iterations allowed
WSS	Real array (K)	output:	the within-cluster sum of squares of each cluster
IFAULT	Integer	output:	see Fault Diagnostics below

Fault Diagnostics

$IFAULT = 0$	No fault
$IFAULT = 1$	At least one cluster is empty after the initial assignment. (A better set of initial cluster centres is called for)
$IFAULT = 2$	The allowed maximum number of iterations is exceeded
$IFAULT = 3$	K is less than or equal to 1 or greater than or equal to M

Auxiliary algorithms

The following auxiliary algorithms are called: *SUBROUTINE OPTRA* (*A, M, N, C, K, IC*1, *IC*2, *NC, AN*1, *AN*2, *NCP, D, ITRAN, LIVE, INDEX*) and *SUBROUTINE QTRAN* (*A, M, N, C, K, IC*1, *IC*2, *NC, AN*1, *AN*2, *NCP, D, ITRAN, INDEX*) which are included.

Related Algorithms

A related algorithm is AS 113 (A transfer algorithm for non-hierarchial classification) given by Banfield and Bassill (1977). This algorithm uses swops as well as transfers to try to overcome the problem of local optima; that is, for all pairs of points, a test is made whether exchanging the clusters to which the points belong will improve the criterion. It will be substantially more expensive than the present algorithm for large M.

The present algorithm is similar to Algorithm AS 58 (Euclidean cluster analysis) given by Sparks (1973). Both algorithms aim at finding a K-partition of the sample, with within-cluster sum of squares which cannot be reduced by moving points from one cluster to the other. However, the implementation of Algorithm AS 58 does not satisfy this condition. At the stage where each point is examined in turn to see if it should be reassigned to a different cluster, only the closest centre is used to check for possible reallocation of the given point; a cluster centre other than the closest one may have the smallest value of the quantity $\{n_l/(n_l+1)\}\, d_l^2$, where n_l is the number of points in cluster l and d_l is the distance from cluster l to the given point. Hence, in general, Algorithm AS 58 does not provide a locally optimal solution.

The two algorithms are tested on various generated data sets. The time consumed on the IBM 370/158 and the within-cluster sum of squares of the resulting K-partitions are given in Table 1. While comparing the entries of the table, note that AS 58 does not give locally optimal solutions and so should be expected to take less time. The *WSS* are different for the two algorithms because they arrive at different partitions of the sets of points. A saving of about 50 per cent in time occurs in *KMNS* due to using "live" sets and due to using a quick-transfer stage which reduces the number of optimal transfer iterations by a factor of 4. Thus, *KMNS* compared to *AS* 58 is locally optimal and takes less time, especially when the number of clusters is large.

Time and Accuracy

The time is approximately equal to $CMNKI$ where I is the number of iterations. For an IBM 370/158, $C = 2{\cdot}1 \times 10^{-5}$ sec. However, different data structures require quite different numbers of iterations; and a careful selection of initial cluster centres will also lead to a considerable saving in time.

Storage requirement: $M(N+3)+K(N+7)$.

TABLE 1

		Time (sec)	*WSS*
1. $M = 1000$, $N = 10$, $K = 10$ (random spherical normal)	AS 58	63·86	7056·71
	KMNS	36·66	7065·59
2. $M = 1000$, $N = 10$, $K = 10$ (two widely separated random normals)	AS 58	43·49	7779·70
	KMNS	19·11	7822·01
3. $M = 1000$, $N = 10$, $K = 50$ (random spherical normal)	AS 58	135·71	4543·82
	KMNS	76·00	4561·48
4. $M = 1000$, $N = 10$, $K = 50$ (two widely separated random normals)	AS 58	95·51	5131·04
	KMNS	57·96	5096·23
5. $M = 50$, $N = 2$, $K = 8$ (two widely separated random normals)	AS 58	0·17	21·03
	KMNS	0·18	21·03

Missing variate values cannot be handled by this algorithm.

The algorithm produces a clustering which is only 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.

The number of iterations required to attain local optimality is usually less than 10.

ADDITIONAL COMMENTS

One way of obtaining the initial cluster centres is suggested here. The points are first ordered by their distances to the overall mean of the sample. Then, for cluster L ($L = 1, 2, \ldots, K$), the $\{1+(L-1)*[M/K]\}$th point is chosen to be its initial cluster centre. In effect, some K sample points are chosen as the initial cluster centres. Using this initialization process, it is guaranteed that no cluster will be empty after the initial assignment in the subroutine. A quick initialization, which is dependent on the input order of the points, takes the first K points as the initial centres.

ACKNOWLEDGEMENTS

This research is supported by National Science Foundation Grant MCS75–08374.

REFERENCES

BANFIELD, C. F. and BASSILL, L. C. (1977). Algorithm AS113. A transfer algorithm for non-hierarchical classification. *Appl. Statist.*, **26**, 206–210.

HARTIGAN, J. A. (1975). *Clustering Algorithms.* New York: Wiley.

SPARKS, D. N. (1973). Algorithm AS 58. Euclidean cluster analysis. *Appl. Statist.*, **22**, 126–130.

```
      SUBROUTINE KMNS(A, M, N, C, K, IC1, IC2, NC, AN1, AN2, NCP,
     *  D, ITRAN, LIVE, ITER, WSS, IFAULT)
C
C        ALGORITHM AS 136  APPL. STATIST. (1979) VOL.28, NO.1
C
C        DIVIDE M POINTS IN N-DIMENSIONAL SPACE INTO K CLUSTERS
C        SO THAT THE WITHIN CLUSTER SUM OF SQUARES IS MINIMIZED.
C
      DIMENSION A(M, N), IC1(M), IC2(M), D(M)
      DIMENSION C(K, N), NC(K), AN1(K), AN2(K), NCP(K)
      DIMENSION ITRAN(K), LIVE(K), WSS(K), DT(2)
C
C        DEFINE BIG TO BE A VERY LARGE POSITIVE NUMBER
C
      DATA BIG /1.0E10/
```

```
      IFAULT = 3
      IF (K .LE. 1 .OR. K .GE. M) RETURN
C
C        FOR EACH POINT I, FIND ITS TWO CLOSEST CENTRES,
C        IC1(I) AND IC2(I).  ASSIGN IT TO IC1(I).
C
      DO 50 I = 1, M
      IC1(I) = 1
      IC2(I) = 2
      DO 10 IL = 1, 2
      DT(IL) = 0.0
      DO 10 J = 1, N
      DA = A(I, J) - C(IL, J)
      DT(IL) = DT(IL) + DA * DA
   10 CONTINUE
      IF (DT(1) .LE. DT(2)) GOTO 20
      IC1(I) = 2
      IC2(I) = 1
      TEMP = DT(1)
      DT(1) = DT(2)
      DT(2) = TEMP
   20 DO 50 L = 3, K
      DB = 0.0
      DO 30 J = 1, N
      DC = A(I, J) - C(L, J)
      DB = DB + DC * DC
      IF (DB .GE. DT(2)) GOTO 50
   30 CONTINUE
      IF (DB .LT. DT(1)) GOTO 40
      DT(2) = DB
      IC2(I) = L
      GOTO 50
   40 DT(2) = DT(1)
      IC2(I) = IC1(I)
      DT(1) = DB
      IC1(I) = L
   50 CONTINUE
C
C        UPDATE CLUSTER CENTRES TO BE THE AVERAGE
C        OF POINTS CONTAINED WITHIN THEM
C
      DO 70 L = 1, K
      NC(L) = 0
      DO 60 J = 1, N
   60 C(L, J) = 0.0
   70 CONTINUE
      DO 90 I = 1, M
      L = IC1(I)
      NC(L) = NC(L) + 1
      DO 80 J = 1, N
   80 C(L, J) = C(L, J) + A(I, J)
   90 CONTINUE
C
C        CHECK TO SEE IF THERE IS ANY EMPTY CLUSTER AT THIS STAGE
C
      IFAULT = 1
      DO 100 L = 1, K
      IF (NC(L) .EQ. 0) RETURN
  100 CONTINUE
      IFAULT = 0
      DO 120 L = 1, K
      AA = NC(L)
      DO 110 J = 1, N
  110 C(L, J) = C(L, J) / AA
C
C        INITIALIZE AN1, AN2, ITRAN AND NCP
C        AN1(L) IS EQUAL TO NC(L) / (NC(L) - 1)
C        AN2(L) IS EQUAL TO NC(L) / (NC(L) + 1)
C        ITRAN(L)=1 IF CLUSTER L IS UPDATED IN THE QUICK-TRANSFER STAGE
C        ITRAN(L)=0 OTHERWISE
C        IN THE OPTIMAL-TRANSFER STAGE, NCP(L) INDICATES THE STEP AT
C        WHICH CLUSTER L IS LAST UPDATED
```

```
C        IN THE QUICK-TRANSFER STAGE, NCP(L) IS EQUAL TO THE STEP AT
C        WHICH CLUSTER L IS LAST UPDATED PLUS M
C
      AN2(L) = AA / (AA + 1.0)
      AN1(L) = BIG
      IF (AA .GT. 1.0) AN1(L) = AA / (AA - 1.0)
      ITRAN(L) = 1
      NCP(L) = -1
  120 CONTINUE
      INDEX = 0
      DO 140 IJ = 1, ITER
C
C        IN THIS STAGE, THERE IS ONLY ONE PASS THROUGH THE DATA.
C        EACH POINT IS REALLOCATED, IF NECESSARY, TO THE CLUSTER
C        THAT WILL INDUCE THE MAXIMUM REDUCTION IN WITHIN-CLUSTER
C        SUM OF SQUARES
C
      CALL OPTRA(A, M, N, C, K, IC1, IC2, NC, AN1, AN2, NCP,
     *  D, ITRAN, LIVE, INDEX)
C
C        STOP IF NO TRANSFER TOOK PLACE IN THE LAST M
C        OPTIMAL-TRANSFER STEPS
C
      IF (INDEX .EQ. M) GOTO 150
C
C        EACH POINT IS TESTED IN TURN TO SEE IF IT SHOULD BE
C        REALLOCATED TO THE CLUSTER WHICH IT IS MOST LIKELY TO
C        BE TRANSFERRED TO (IC2(I)) FROM ITS PRESENT CLUSTER (IC1(I)).
C        LOOP THROUGH THE DATA UNTIL NO FURTHER CHANGE IS TO TAKE PLACE
C
      CALL QTRAN(A, M, N, C, K, IC1, IC2, NC, AN1, AN2,
     *  NCP, D, ITRAN, INDEX)
C
C        IF THERE ARE ONLY TWO CLUSTERS,
C        NO NEED TO RE-ENTER OPTIMAL-TRANSFER STAGE
C
      IF (K .EQ. 2) GOTO 150
C
C        NCP HAS TO BE SET TO 0 BEFORE ENTERING OPTRA
C
      DO 130 L = 1, K
  130 NCP(L) = 0
  140 CONTINUE
C
C        SINCE THE SPECIFIED NUMBER OF ITERATIONS IS EXCEEDED
C        IFAULT IS SET TO BE EQUAL TO 2.
C        THIS MAY INDICATE UNFORESEEN LOOPING
C
      IFAULT = 2
C
C        COMPUTE WITHIN CLUSTER SUM OF SQUARES FOR EACH CLUSTER
C
  150 DO 160 L = 1, K
      WSS(L) = 0.0
      DO 160 J = 1, N
      C(L, J) = 0.0
  160 CONTINUE
      DO 170 I = 1, M
      II = IC1(I)
      DO 170 J = 1, N
      C(II, J) = C(II, J) + A(I, J)
  170 CONTINUE
      DO 190 J = 1, N
      DO 180 L = 1, K
  180 C(L, J) = C(L, J) / FLOAT(NC(L))
      DO 190 I = 1, M
      II = IC1(I)
      DA = A(I, J) - C(II, J)
      WSS(II) = WSS(II) + DA * DA
  190 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OPTRA(A, M, N, C, K, IC1, IC2, NC, AN1,
     *  AN2, NCP, D, ITRAN, LIVE, INDEX)
C
C       ALGORITHM AS 136.1  APPL. STATIST. (1979) VOL.28, NO.1
C
C       THIS IS THE OPTIMAL-TRANSFER STAGE
C
C       EACH POINT IS REALLOCATED, IF NECESSARY, TO THE
C       CLUSTER THAT WILL INDUCE A MAXIMUM REDUCTION IN
C       THE WITHIN-CLUSTER SUM OF SQUARES
C
      DIMENSION A(M, N), IC1(M), IC2(M), D(M)
      DIMENSION C(K, N), NC(K), AN1(K), AN2(K), NCP(K)
      DIMENSION ITRAN(K), LIVE(K)
C
C       DEFINE BIG TO BE A VERY LARGE POSITIVE NUMBER
C
      DATA BIG /1.0E10/
C
C       IF CLUSTER L IS UPDATED IN THE LAST QUICK-TRANSFER STAGE,
C       IT BELONGS TO THE LIVE SET THROUGHOUT THIS STAGE.
C       OTHERWISE, AT EACH STEP, IT IS NOT IN THE LIVE SET IF IT
C       HAS NOT BEEN UPDATED IN THE LAST M OPTIMAL-TRANSFER STEPS
C
      DO 10 L = 1, K
      IF (ITRAN(L) .EQ. 1) LIVE(L) = M + 1
   10 CONTINUE
      DO 100 I = 1, M
      INDEX = INDEX + 1
      L1 = IC1(I)
      L2 = IC2(I)
      LL = L2
C
C       IF POINT I IS THE ONLY MEMBER OF CLUSTER L1, NO TRANSFER
C
      IF (NC(L1) .EQ. 1) GOTO 90
C
C       IF L1 HAS NOT YET BEEN UPDATED IN THIS STAGE
C       NO NEED TO RECOMPUTE D(I)
C
      IF (NCP(L1) .EQ. 0) GOTO 30
      DE = 0.0
      DO 20 J = 1, N
      DF = A(I, J) - C(L1, J)
      DE = DE + DF * DF
   20 CONTINUE
      D(I) = DE * AN1(L1)
C
C       FIND THE CLUSTER WITH MINIMUM R2
C
   30 DA = 0.0
      DO 40 J = 1, N
      DB = A(I, J) - C(L2, J)
      DA = DA + DB * DB
   40 CONTINUE
      R2 = DA * AN2(L2)
      DO 60 L = 1, K
C
C       IF I IS GREATER THAN OR EQUAL TO LIVE(L1), THEN L1 IS
C       NOT IN THE LIVE SET.  IF THIS IS TRUE, WE ONLY NEED TO
C       CONSIDER CLUSTERS THAT ARE IN THE LIVE SET FOR POSSIBLE
C       TRANSFER OF POINT I.  OTHERWISE, WE NEED TO CONSIDER
C       ALL POSSIBLE CLUSTERS
C
      IF (I .GE. LIVE(L1) .AND. I .GE. LIVE(L) .OR.
     *  L .EQ. L1 .OR. L .EQ. LL) GOTO 60
      RR = R2 / AN2(L)
      DC = 0.0
      DO 50 J = 1, N
      DD = A(I, J) - C(L, J)
      DC = DC + DD * DD
      IF (DC .GE. RR) GOTO 60
```

```
   50 CONTINUE
      R2 = DC * AN2(L)
      L2 = L
   60 CONTINUE
      IF (R2 .LT. D(I)) GOTO 70
C
C        IF NO TRANSFER IS NECESSARY, L2 IS THE NEW IC2(I)
C
      IC2(I) = L2
      GOTO 90
C
C        UPDATE CLUSTER CENTRES, LIVE, NCP, AN1 AND AN2
C        FOR CLUSTERS L1 AND L2, AND UPDATE IC1(I) AND IC2(I)
C
   70 INDEX = 0
      LIVE(L1) = M + I
      LIVE(L2) = M + I
      NCP(L1) = I
      NCP(L2) = I
      AL1 = NC(L1)
      ALW = AL1 - 1.0
      AL2 = NC(L2)
      ALT = AL2 + 1.0
      DO 80 J = 1, N
      C(L1, J) = (C(L1, J) * AL1 - A(I, J)) / ALW
      C(L2, J) = (C(L2, J) * AL2 + A(I, J)) / ALT
   80 CONTINUE
      NC(L1) = NC(L1) - 1
      NC(L2) = NC(L2) + 1
      AN2(L1) = ALW / AL1
      AN1(L1) = BIG
      IF (ALW .GT. 1.0) AN1(L1) = ALW / (ALW - 1.0)
      AN1(L2) = ALT / AL2
      AN2(L2) = ALT / (ALT + 1.0)
      IC1(I) = L2
      IC2(I) = L1
   90 CONTINUE
      IF (INDEX .EQ. M) RETURN
  100 CONTINUE
      DO 110 L = 1, K
C
C        ITRAN(L) IS SET TO ZERO BEFORE ENTERING QTRAN.
C        ALSO, LIVE(L) HAS TO BE DECREASED BY M BEFORE
C        RE-ENTERING OPTRA
C
      ITRAN(L) = 0
      LIVE(L) = LIVE(L) - M
  110 CONTINUE
      RETURN
      END
C
      SUBROUTINE QTRAN(A, M, N, C, K, IC1, IC2, NC, AN1,
     *  AN2, NCP, D, ITRAN, INDEX)
C
C        ALGORITHM AS 136.2  APPL. STATIST. (1979) VOL.28, NO.1
C
C        THIS IS THE QUICK TRANSFER STAGE.
C        IC1(I)  IS THE CLUSTER WHICH POINT I BELONGS TO.
C        IC2(I)  IS THE CLUSTER WHICH POINT I IS MOST
C        LIKELY TO BE TRANSFERRED TO.
C        FOR EACH POINT I, IC1(I) AND IC2(I) ARE SWITCHED, IF
C        NECESSARY, TO REDUCE WITHIN CLUSTER SUM OF SQUARES.
C        THE CLUSTER CENTRES ARE UPDATED AFTER EACH STEP
C
      DIMENSION A(M, N), IC1(M), IC2(M), D(M)
      DIMENSION C(K, N), NC(K), AN1(K), AN2(K), NCP(K), ITRAN(K)
C
C        DEFINE BIG TO BE A VERY LARGE POSITIVE NUMBER
C
      DATA BIG /1.0E10/
```

```
C        IN THE OPTIMAL-TRANSFER STAGE, NCP(L) INDICATES THE
C        STEP AT WHICH CLUSTER L IS LAST UPDATED
C        IN THE QUICK-TRANSFER STAGE, NCP(L) IS EQUAL TO THE
C        STEP AT WHICH CLUSTER L IS LAST UPDATED PLUS M
C
      ICOUN = 0
      ISTEP = 0
   10 DO 70 I = 1, M
      ICOUN = ICOUN + 1
      ISTEP = ISTEP + 1
      L1 = IC1(I)
      L2 = IC2(I)
C
C        IF POINT I IS THE ONLY MEMBER OF CLUSTER L1, NO TRANSFER
C
      IF (NC(L1) .EQ. 1) GOTO 60
C
C        IF ISTEP IS GREATER THAN NCP(L1), NO NEED TO RECOMPUTE
C        DISTANCE FROM POINT I TO CLUSTER L1
C        NOTE THAT IF CLUSTER L1 IS LAST UPDATED EXACTLY M STEPS
C        AGO WE STILL NEED TO COMPUTE THE DISTANCE FROM POINT I
C        TO CLUSTER L1
C
      IF (ISTEP .GT. NCP(L1)) GOTO 30
      DA = 0.0
      DO 20 J = 1, N
      DB = A(I, J) - C(L1, J)
      DA = DA + DB * DB
   20 CONTINUE
      D(I) = DA * AN1(L1)
C
C        IF ISTEP IS GREATER THAN OR EQUAL TO BOTH NCP(L1) AND
C        NCP(L2) THERE WILL BE NO TRANSFER OF POINT I AT THIS STEP
C
   30 IF (ISTEP .GE. NCP(L1) .AND. ISTEP .GE. NCP(L2)) GOTO 60
      R2 = D(I) / AN2(L2)
      DD = 0.0
      DO 40 J = 1, N
      DE = A(I, J) - C(L2, J)
      DD = DD + DE * DE
      IF (DD .GE. R2) GOTO 60
   40 CONTINUE
C
C        UPDATE CLUSTER CENTRES, NCP, NC, ITRAN, AN1 AND AN2
C        FOR CLUSTERS L1 AND L2.  ALSO, UPDATE IC1(I) AND IC2(I).
C        NOTE THAT IF ANY UPDATING OCCURS IN THIS STAGE,
C        INDEX IS SET BACK TO 0
C
      ICOUN = 0
      INDEX = 0
      ITRAN(L1) = 1
      ITRAN(L2) = 1
      NCP(L1) = ISTEP + M
      NCP(L2) = ISTEP + M
      AL1 = NC(L1)
      ALW = AL1 - 1.0
      AL2 = NC(L2)
      ALT = AL2 + 1.0
      DO 50 J = 1, N
      C(L1, J) = (C(L1, J) * AL1 - A(I, J)) / ALW
      C(L2, J) = (C(L2, J) * AL2 + A(I, J)) / ALT
   50 CONTINUE
      NC(L1) = NC(L1) - 1
      NC(L2) = NC(L2) + 1
      AN2(L1) = ALW / AL1
      AN1(L1) = BIG
      IF (ALW .GT. 1.0) AN1(L1) = ALW / (ALW - 1.0)
      AN1(L2) = ALT / AL2
      AN2(L2) = ALT / (ALT + 1.0)
      IC1(I) = L2
      IC2(I) = L1
C
C        IF NO REALLOCATION TOOK PLACE IN THE LAST M STEPS, RETURN
C
   60 IF (ICOUN .EQ. M) RETURN
   70 CONTINUE
      GOTO 10
      END
```