```
C        COMPUTE THE CONTRIBUTION FROM THE MARGINS
C
   30 IFT = 1
      IF (TOTAL .GT. 100) RETURN
      DO 40 I = 1, 100
   40 NLOGN(I) = FLOAT(I) * PLOG(FLOAT(I))
      DEV = ZERO
      IF (MODEL .LT. 0) GOTO 90
      DO 50 I = 1, M
      J = RMARGN(I)
      DEV = DEV - NLOGN(J)
   50 CONTINUE
      IF (MODEL .EQ. 0) GOTO 90
      DO 80 J = 1, N
      I = CMARGN(J)
      DEV = DEV - NLOGN(I)
   80 CONTINUE
      GOTO 100
   90 DEV = DEV + FLOAT(TOTAL) * PLOG(FLOAT(N) * FLOAT(M ** (-MODEL)))
  100 DEVMAR = DEV + NLOGN(TOTAL) * FLOAT(MODEL)
      GOTO 10
      END
```

Algorithm AS 113

A Transfer Algorithm for Non-hierarchical Classification

By C. F. Banfield and L. C. Bassill

Rothamsted Experimental Station, Harpenden, Herts, Britain

Keywords: NON-HIERARCHICAL CLASSIFICATION; TRANSFER ALGORITHM; ITERATIVE RELOCATION

Language

ISO Fortran

Purpose

Non-hierarchical classification can be used to partition a sample of n objects (for each of which the same v variates are measured) into k more or less homogeneous classes, which it is hoped will correspond to natural populations or to marked features of the sample. The classification is obtained by optimizing some suitable criterion and is non-hierarchical because the classes need not combine to form nested classes as their number is reduced. Several criteria to be optimized have been suggested, such as maximizing the sums of squares between the classes (see Hartigan, 1975, for example), minimizing the determinant of the pooled within-class dispersion matrix (Friedman and Rubin, 1967) or maximizing the number of agreements with binary class-predictors (Gower, 1974). Which criterion should be used depends on the nature of the problem and the form of the data, but all such criteria can be optimized using a transfer algorithm.

Hartigan (1975) suggests several starting, movement and updating options for a transfer algorithm but, unlike this algorithm, does not consider the swopping of objects between classes or give a general algorithm for several criteria. The algorithm given here is used by the Genstat program (Nelder *et al.*, 1975).

Method

The algorithm has two distinct phases—one of transferring an object from one class to another and the other of swopping two objects between classes. Given a starting classification, each possible transfer is tested in turn to see if it would improve the value of the criterion, and if beneficial it is executed. Transfers from classes with only one member are not permitted. When no further transfers can improve the criterion value, each possible swop of two objects

between different classes is tested, and again these are executed only if found to be beneficial. When no further swops give an improvement the transfer phase is re-entered, and so on until no more transfers or swops can improve the value of the criterion.

Description

Given a classification of the objects in *CLASS*(*IN*) with the corresponding class sizes in *CLSIZE*(*IK*), and the criterion value in *CRITVL*, the subroutine *TRNSFR* will systematically calculate the class increments to the criterion value for all possible transfers of an object *I* from class *M* to class *L*. If a particular transfer will improve the criterion value it will execute the transfer and update *CRITVL*, *CLSIZE*(*IK*) and *CLASS*(*IN*) accordingly. When no more transfers are beneficial *TRNSFR* returns control to the main program which then calls *SWOP*. This subroutine calculates class increments for all possible swops of objects *J* from class *L* to *M* and *I* from class *M* to *L* and executes those that improve the criterion value. When no more swops are beneficial, control is returned to the main program. *TRNSFR* is then recalled.

Before *TRNSFR* is called for the first time a starting classification must be provided in *CLASS*(*IN*), and *CLSIZE*(*IK*) and *CRITVL* initialized accordingly. If the value of *NTRANS* (holding the number of executed transfers or swops) is zero when leaving *TRNSFR*, and *SWOP* has been entered at least once, or if it is zero when leaving *SWOP*, no further improvement may be obtained by these methods.

Structure

SUBROUTINE TRNSFR (*VARVAL, CLASS, CLSIZE, IN, IK, IV, CRITVL, NTRANS, IFAULT*)

Formal parameters

VARVAL	Real array (*IN*, *IV*)	input:	the variate values
CLASS	Integer array (*IN*)	input/ output:	the current classification of each object
CLSIZE	Integer array (*IK*)	input/ output:	the current number of objects in each class
IN	Integer	input:	the number n of objects
IK	Integer	input:	the number k of classes
IV	Integer	input:	the number v of variates
CRITVL	Real	input/ output:	the current value of the criterion
NTRANS	Integer	output:	the number of executed transfers
IFAULT	Integer	output:	a fault indicator, equal to: 0 no fault 1 $IK \leqslant 1$ 2 $IN \leqslant IK$

SUBROUTINE SWOP (*VARVAL, CLASS, CLSIZE, IN, IK, IV, CRITVL, NTRANS, IFAULT*)

Formal parameters

As for *TRNSFR*, except

NTRANS	Integer	output:	the number of executed swops

For both subroutines a preset tolerance parameter, *EPS*, must be given in a DATA statement for use when testing if increments are equal to zero.

Auxiliary Algorithms

TRNSFR calls subroutine *CRTRAN* (*VARVAL, CLSIZE, IN, IK, IV, I, L, M, ISWICH, INC*) for two purposes indicated by the value of *ISWICH*. When *ISWICH* equals 1 *CRTRAN*

will calculate the increment, *INC*, to the criterion value if object *I* were to be transferred from class *M* to *L*. If this increment would improve the criterion value, *ISWICH* is put equal to 2 and *CRTRAN* then updates all values to the chosen criterion for that transfer. Similarly *SWOP* calls subroutine *CRSWOP* (*VARVAL, CLSIZE, IN, IK, IV, I, J, L, M, ISWICH, INC*) which, again depending on the value of *ISWICH*, either calculates the increment to the criterion value or updates relevant values, if objects *I* and *J* are swopped between classes *L* and *M*. Both *CRTRAN* and *CRSWOP* must be provided by the user for the particular criterion to be optimized by the transfer algorithm, and as such they will usually require extra parameters referring to arrays that hold values relevant to the chosen criterion.

For instance, if the sum of squares between classes is to be optimized (i.e. the sum of squares within classes is minimized) additional arrays must be specified to hold the sums of squares for each variate within a class and the mean variate values for each class. These arrays will then be updated in *CRTRAN* and *CRSWOP* when a particular transfer or swop has been found to improve the criterion value. If a transfer is executed then for the jth variate ($j = 1, \ldots, v$):

$$S'_{Mj} = S_{Mj} - n(x_{Ij} - \bar{x}_{Mj})^2/(n-1) \quad \text{and} \quad \bar{x}'_{Mj} = (n\bar{x}_{Mj} - x_{Ij})/(n-1),$$

where S_{Mj} and S'_{Mj} are the sums of squares within class M before and after object I is transferred to class L, n is the number of objects in class M before I is transferred, x_{Ij} is the value of the jth variate for object I, and $\bar{x}_{Mj}$ and $\bar{x}'_{Mj}$ are the means of the jth variate values for the objects in class M before and after I is transferred. Correction S'_{Lj} and $\bar{x}'_{Lj}$ are also necessary for the class L.

Similarly, if the chosen criterion is to minimize the determinant of the pooled within-class dispersion matrix, an extra array must be specified for *CRTRAN* and *CRSWOP* to hold initially the dispersion matrix and then its current inverse.

$$|\mathbf{W} + \mathbf{aa}' - \mathbf{bb}'| = |\mathbf{W}|\{(1 - \mathbf{a}'\mathbf{W}^{-1}\mathbf{a})(1 + \mathbf{b}'\mathbf{W}^{-1}\mathbf{b}) + (\mathbf{b}'\mathbf{W}^{-1}\mathbf{a})^2\} \tag{1}$$

$$(\mathbf{W} + \mathbf{aa}')^{-1} = \mathbf{W}^{-1} - (\mathbf{W}^{-1}\mathbf{a})(\mathbf{W}^{-1}\mathbf{a})'/(1 + \mathbf{a}'\mathbf{W}^{-1}\mathbf{a}) \tag{2}$$

and

$$(\mathbf{W} + \mathbf{aa}' - \mathbf{bb}')^{-1} = (\mathbf{W} + \mathbf{aa}')^{-1} + \{(\mathbf{W} + \mathbf{aa}')^{-1}\mathbf{b}\}\{(\mathbf{W} + \mathbf{aa}')^{-1}\mathbf{b}\}'/\{1 - \mathbf{b}'(\mathbf{W} + \mathbf{aa}')^{-1}\mathbf{b}\} \tag{3}$$

where $\mathbf{W}$ is the dispersion matrix, and $\mathbf{aa}'$ and $\mathbf{bb}'$ are the increments to $\mathbf{W}$ for the changes in classes L and M respectively when a transfer is executed. For the jth variate:

$$a_j = \sqrt{n_L}(x_{Ij} - \bar{x}_{Lj})/\sqrt{\{(n_L + 1)(N - K)\}} \quad \text{and} \quad b_j = \sqrt{n_M}(x_{Ij} - \bar{x}_{Mj})/\sqrt{\{(n_M - 1)(N - K)\}}$$

where n_L and n_M are the sizes of classes L and M before object I is transferred from class M to L, N is the total number of objects and K is the total number of classes. Equation (1) shows how the determinant is incremented in terms of the inverse, and (2) and (3) show how the inverse is incremented. A swop can be treated as two transfers if temporary storage is provided for the updated values after the first of the transfers.

Accuracy

The optimum classification found may not be unique as other classifications may give the same criterion value. Also it may not be a global optimum, so different starting classifications should be tried. Using the solution to a classification with greater than k classes as the starting classification for one with fewer classes and working down to the desired k-classification can help avoid local optima.

Whether missing variate values can be handled by the transfer algorithm depends solely on the criterion chosen and the auxiliary algorithms *CRTRAN* and *CRSWOP*.

Acknowledgements

We wish to thank Mr J. C. Gower and the referee for their many helpful suggestions.

REFERENCES

FRIEDMAN, H. P. and RUBIN, J. (1967). On some invariant criteria for grouping data. *J. Amer. Statist. Ass.*, **62**, 1159–1186.

GOWER, J. C. (1974). Maximal predictive classification. *Biometrics*, **30**, 643–654.

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

NELDER, J. A. and members of the ROTHAMSTED STATISTICS DEPARTMENT (1975). *Genstat Reference Manual*. Program Library Unit, Edinburgh Regional Computer Centre.

```
      SUBROUTINE TRNSFR(VARVAL, CLASS, CLSIZE, IN, IK, IV, CRITVL,
     *  NTRANS, IFAULT)
C
C        ALGORITHM AS 113.1  APPL. STATIST. (1977) VOL. 26, NO. 2
C
C        TRANSFERS OBJECTS FROM ONE CLASS TO ANOTHER
C        IF THIS IMPROVES THE VALUE OF THE CRITERION
C
      REAL VARVAL(IN, IV)
      REAL INC, INCO
      INTEGER CLASS(IN), CLSIZE(IK)
C
C        DEFINE EPS, A PRESET TOLERANCE PARAMETER
C
      DATA EPS /1.0E-38/
C
      IFAULT = 1
      IF (IK .LE. 1) RETURN
      IFAULT = 2
      IF (IN .LE. IK) RETURN
      IFAULT = 0
      NTRANS = 0
    1 I = 0
      ICOUNT = 0
    2 I = I + 1
      IF (ICOUNT .GE. IN) RETURN
      IF (I .GT. IN) GOTO 1
      M = CLASS(I)
      IF (CLSIZE(M) .GT. 1) GOTO 3
      ICOUNT = ICOUNT + 1
      GOTO 2
    3 INCO = -EPS
      LO = M
      DO 4 L = 1, IK
C
C        TEST THE TRANSFER OF OBJECT I FROM CLASS M TO CLASS L
C
      IF (L .EQ. M) GOTO 4
      ISWICH = 1
      CALL CRTRAN(VARVAL, CLSIZE, IN, IK, IV, I, L, M, ISWICH, INC)
      IF (INC .GE. INCO) GOTO 4
C
C        REMEMBER VALUES OF L AND INC
C
      LO = L
      INCO = INC
    4 CONTINUE
      ICOUNT = ICOUNT + 1
      IF (LO .EQ. M) GOTO 2
C
C        EXECUTE THE TRANSFER OF OBJECT I FROM CLASS M TO CLASS LO
C
      L = LO
      CRITVL = CRITVL + INCO
      ICOUNT = 0
      ISWICH = 2
      CALL CRTRAN(VARVAL, CLSIZE, IN, IK, IV, I, L, M, ISWICH, INC)
      NTRANS = NTRANS + 1
      CLASS(I) = L
      CLSIZE(L) = CLSIZE(L) + 1
      CLSIZE(M) = CLSIZE(M) - 1
      GOTO 2
      END
```

```
      SUBROUTINE SWOP(VARVAL, CLASS, CLSIZE, IN, IK, IV, CRITVL, NTRANS,
     *  IFAULT)
C
C        ALGORITHM AS 113.2  APPL. STATIST. (1977) VOL. 26, NO. 2
C
C        SWOPS OBJECTS BETWEEN CLASSES IF THIS
C        IMPROVES THE VALUE OF THE CRITERION
C
      REAL VARVAL(IN, IV)
      REAL INC
      INTEGER CLASS(IN), CLSIZE(IK)
C
C        DEFINE EPS, A PRESET TOLERANCE PARAMETER
C
      DATA EPS /1.0E-38/
C
      IFAULT = 1
      IF (IK .LE. 1) RETURN
      IFAULT = 2
      IF (IN .LE. IK) RETURN
      IFAULT = 0
      ICOUNT = 0
      NTRANS = 0
      ITOP = IN * (IN - 1) / 2
    1 I = 1
    2 I = I + 1
      IF (ICOUNT .GE. ITOP) RETURN
      IF (I .GT. IN) GOTO 1
      L = CLASS(I)
      K = L
      IT = I - 1
      DO 3 J = 1, IT
C
C        TEST THE SWOP OF OBJECT I FROM CLASS
C        M TO L AND OBJECT J FROM CLASS L TO M
C
      ICOUNT = ICOUNT + 1
      M = CLASS(J)
      IF (L .EQ. M) GOTO 3
      IF (CLSIZE(L) .EQ. 1 .AND. CLSIZE(M) .EQ. 1) GOTO 3
      ISWICH = 1
      CALL CRSWOP(VARVAL, CLSIZE, IN, IK, IV, I, J, L, M, ISWICH, INC)
      IF (INC .GE. -EPS) GOTO 3
C
C        EXECUTE THE SWOP OF OBJECT I FROM CLASS
C        M TO L AND OBJECT J FROM CLASS L TO M
C
      CRITVL = CRITVL + INC
      ICOUNT = 0
      ISWICH = 2
      CALL CRSWOP(VARVAL, CLSIZE, IN, IK, IV, I, J, L, M, ISWICH, INC)
      NTRANS = NTRANS + 1
      CLASS(I) = M
      CLASS(J) = L
      L = M
    3 CONTINUE
      GOTO 2
      END
```