Nonhierarchical clustering technique (GROUPER)*

RONALD G. SHERWIN†
Department of International Relations

and

NIEN-LING WAYMAN
University Computer Center
University of Southern California
Los Angeles, California 90007

A technique for reducing a larger interaction group to its underlying subgroups has been described in the literature of sociometry, and is adapted here for computer use (Beum & Brundage, 1950). The technique can be employed in cases where assumptions about variable intercorrelation or the effects of transforming data are undesirable. The technique involves the creation of a sociomatrix. The cell entries of such a matrix can be of any value and of any significance so long as they measure relations among the units of a group. One convention is to have cell entries signify action structures among group members where any ij entry equals the number of actions targeted from the units designated as the ith row to the unit designated as the jth column. Where no action was targeted from one unit to another—i.e., where no interunit relations existed—a zero would appear in the appropriate ij cell.

The Algorithm. The problem is one of systematically rearranging the rows and columns of the matrix so that zero entries move away from the diagonal and nonzero entries cluster about the diagonal. Doing this should reveal unit clusters where the concentration of interunit relations is highest. The process involves the following steps: (1) Sum each column to obtain column sums. (2) Assign a weight to each row so that the first row's weight is equal to N, the number of rows in the matrix; the second row's weight is equal to N – 1; the third row's weight is equal to N – 2; and so on. The last row will have a weight of 1. (3) Obtain the weighted sum of each column by multiplying each column entry by its corresponding row weight, and then summing the weighted entries for each column. (4) Obtain weighted averages for each column by dividing each column's weighted sum by its column sum. (5) Rank the weighted averages from N for the largest through 1 for the smallest. (6) Rearrange the rows and columns of the matrix so that the column with the highest weighted average ranking becomes Column 1 and its corresponding row becomes Row 1; then the column with the second highest weighted average ranking becomes Column 2 and its corresponding row becomes Row 2; and so on through N. These six steps are repeated until no more row-column shifting occurs.

After a number of iterations, if the larger group does indeed contain subgroups, subgroups of rows and columns will begin to occupy the same, or near the same, position in the row column rankings of subsequent matrices. Any row-column shifting which may occur in subsequent iterations will be intrasubgroup shifting rather than intersubgroup shifting.

In practice, it has been found that the number of iterations required to delineate subgroups, and the clarity with which subgroups are defined, are dependent on the larger group's structure. In cases where the larger group's structure contains subgroups with few intergroup relations, the subgroups will be quickly and clearly defined by a few iterations. On the other hand, a large number of iterations will reveal cases where the larger group does not contain distinct subgroups.

This procedure can be useful in many areas, as the following examples illustrate.

Example 1. The international interaction for 1 year among 142 nations was coded from *The New York Times*, and the total interaction from each nation to the next was recorded on a sociomatrix. The above process was performed on the resulting matrix 142 times to reveal distinct clusters of Middle Eastern, Asian, Latin American, European, and African nations (Sherwin, 1972).

Example 2. Communication flow between the different departments of a larger institution have been depicted on a sociomatrix and analyzed with the above algorithm to help architects plan floor layouts for multistory buildings.

Example 3. The interflight connections of passengers and baggage might be analyzed to aid in the design of passenger and baggage handling facilities at large airports.

Programming Techniques. The main problem in programming the above algorithm is that it requires relatively large amounts of machine time and core area. This problem has been alleviated as the following discussion demonstrates. Consider the 5 x 5 matrix in Fig. 1. Now, by inserting a first row and a first column

$$\text{Matrix A} \quad \begin{array}{|ccccc|} \hline 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ \hline \end{array}$$

Fig. 1. Matrix A, 5 x 5.

*This research was supported by ONR-ARPA Contract N00014-67-A-0296-0004.

†Now at the Institute of International Studies, University of California, Berkeley, California 94720.

$$\begin{array}{c|ccccc} & \text{I} & \text{II} & \text{III} & \text{IV} & \text{V} \\ \hline \text{I} & 1 & 2 & 3 & 4 & 5 \\ \text{II} & 6 & 7 & 8 & 9 & 10 \\ \text{III} & 11 & 12 & 13 & 14 & 15 \\ \text{IV} & 16 & 17 & 18 & 19 & 20 \\ \text{V} & 21 & 22 & 23 & 24 & 25 \end{array}$$

Fig. 2. Matrix A, 6 x 6.

$$\begin{array}{c|ccccc} & \text{V} & \text{IV} & \text{III} & \text{II} & \text{I} \\ \hline \text{I} & 1 & 2 & 3 & 4 & 5 \\ \text{II} & 6 & 7 & 8 & 9 & 10 \\ \text{III} & 11 & 12 & 13 & 14 & 15 \\ \text{IV} & 16 & 17 & 18 & 19 & 20 \\ \text{V} & 21 & 22 & 23 & 24 & 25 \end{array}$$

Fig. 3. Matrix A, first row interchanged.

$$\begin{array}{c|ccccc} & \text{V} & \text{VI} & \text{III} & \text{II} & \text{I} \\ \hline \text{I} & 5 & 2 & 3 & 4 & 1 \\ \text{II} & 10 & 7 & 8 & 9 & 6 \\ \text{III} & 15 & 12 & 13 & 14 & 11 \\ \text{IV} & 20 & 17 & 18 & 19 & 16 \\ \text{V} & 25 & 22 & 23 & 24 & 21 \\ & \uparrow & & & & \uparrow \end{array}$$

Fig. 4. Matrix A, columns interchanged.

$$\begin{array}{c|ccccc|c} & \text{V} & \text{IV} & \text{III} & \text{II} & \text{I} & \\ \hline \text{V} & 25 & 22 & 23 & 24 & 21 & \leftarrow \\ \text{II} & 10 & 7 & 8 & 9 & 6 & \\ \text{III} & 15 & 12 & 13 & 14 & 11 & \\ \text{IV} & 20 & 17 & 18 & 19 & 16 & \\ \text{I} & 5 & 2 & 3 & 4 & 1 & \leftarrow \end{array}$$

Fig. 5. Matrix A, rows interchanged.

as the identification (ID) of the columns and rows, respectively, Matrix A is 6 x 6, as in Fig. 2. To perform Steps 1 through 5 described above, the ranks for the columns are obtained and stored in an array called RAT. In this case, the values for RAT(2), RAT(3), . . ., RAT(6) are 1, 2, 3, 4, and 5. Step 6 requires a total rearrangement of the matrix. To save time and core, RAT is first sorted in descending order, and at the same time the elements in the first row are moved accordingly. The programming statements are:

```
         RAT(1)=0
         DO 60 I=3,NP        (NP = no. of rows + 1)
         J=I
70       IF (RAT(J).LE.RAT(J-1)) GO TO 60
         XSAVE = RAT(J)
         RAT(J)=RAT(J-1)
         RAT(J-1)=XSAVE
         SAVE=A(1,J)
         A(1,J)=A(1,J-1)
         A(1,J-1)=SAVE
         IF ((J-1).EQ.2) GO TO 60
         J=J-1
         GO TO 70
60       CONTINUE
           .
           .
           .
```

The resultant values of RAT(2), RAT(3), . . ., RAT(6) are 5, 4, 3, 2, and 1, and Matrix A appears as in Fig. 3. (NOTE: Only the elements in the first row–the ID row–have been interchanged.)

Step A. Each element in Row 1 (i.e., the Column ID) is compared with elements of Column 1 (i.e., Row ID) until they contain the same ID. For example, A(1,2) is compared with each element of Column 1 (except the first one) (i.e., A(2,1), A(3,1), . . ., A(6,1) until it finds A(1,2) and A(6,1), which both contain the ID, V; then the elements of Column 2 (except the first element) are interchanged with the elements of Column 6. Matrix A now appears as in Fig. 4. (NOTE: Arrows point to the columns or rows that have been interchanged.)

Step B. Then the relative rows are interchanged, as in Fig. 5 (i.e., Row 2 and Row 6; they have IDs I and V). Next, the third element of Row 1 [i.e., A(1,3)] is compared with the rest of Column 1 [i.e., A(3,1),

Step A:

$$\begin{array}{c|ccccc} & \text{V} & \text{IV} & \text{III} & \text{II} & \text{I} \\ \hline \text{V} & 25 & 24 & 23 & 22 & 21 \\ \text{II} & 10 & 9 & 8 & 7 & 6 \\ \text{III} & 15 & 14 & 13 & 12 & 11 \\ \text{IV} & 20 & 19 & 18 & 17 & 16 \\ \text{I} & 5 & 4 & 3 & 2 & 1 \\ & & \uparrow & & \uparrow & \end{array}$$

Step B:

$$\begin{array}{c|ccccc|c} & \text{V} & \text{IV} & \text{III} & \text{II} & \text{I} & \\ \hline \text{V} & 25 & 24 & 23 & 22 & 21 & \\ \text{IV} & 20 & 19 & 18 & 17 & 16 & \leftarrow \\ \text{III} & 15 & 14 & 13 & 12 & 11 & \\ \text{II} & 10 & 9 & 8 & 7 & 6 & \leftarrow \\ \text{I} & 5 & 4 & 3 & 2 & 1 & \end{array}$$

Fig. 6. Matrix A, additional row/column interchanges.

	J	A	E	F	C	I	B	G	H	D
J	0		5	5		2		3		
A		0			4		3			5
E	5		0	2		2		3	1	
F	5		2	0				3	4	
C		4			0		3			6
I			2			0			1	
B					3		0			1
G			3	3		6		0	2	
H				4				2	0	
D		5			6					0

Fig. 7. 10 x 10 matrix before technique is applied.

	F	E	I	G	H	J	C	A	D	B
F	0	2		3	4	5				
E	2	0	2	3	1	5				
I		2	0		1					
G	3	3	6	0	2					
H	4			2	0					
J	5	5	2	3		0				
C							0	4	6	3
A							4	0	5	3
D							6	5	0	
B							3		1	0

Fig. 8. 10 x 10 matrix after technique is applied.

A(4,1), A(5,1), and A(6,1)], but not the second element of Column 1, because this has already been matched. In this case, A(5,1) has the same ID value as A(1,3). So Columns 3 and 5 are interchanged; likewise, Rows 3 and 5 are interchanged (see Fig. 6). By examining A(1,4) and A(4,1), it is found that they both have ID III. Therefore, no movement is necessary. Similarly, it is not necessary to switch the remaining rows and columns. This process can be carried out many times, depending on the parameters given by the user. The programming statements for Steps A and B are as follows:

```
      DO 100 J=2,NP
      DO 110 I=J,NP
      KKI=I
C     CHECK ROW AND COLUMN ID
      IF(A(I,1).EQ.A(1,J)) GO TO 115
110   CONTINUE
      GO TO 100
C     CHECK IF THE ROW AND COLUMN HAS THE SAME POSITION
115   IF (KKI.EQ.J) GO TO 100
      DO 116 I=2,NP
C     DURING THE PROCESS OF COLUMN INTERCHANGING, CHECK
C     IF THE ELEMENTS HAVE THE SAME VALUE
      IF (A(I,J).EQ.A(I,KII)) GO TO 116
      SAVE=A(I,J)
      A(I,J)=A(I,KII)
      A(I,KII)=SAVE
116   CONTINUE
C     ROWS INTERCHANGING
      DO 230 K=1,NP
      SAVE=A(J,K)
      A(J,K)=A(KKI,K)
      A(KKI,K)=SAVE
230   CONTINUE
100   CONTINUE
      GO TO 52
```

For an n x n matrix, this method requires the interchanging of no more than 2x(n – 1) rows and 2x(n – 1) columns. Also, this method requires only two elements of intermediate work core which have been called XSAVE and SAVE.

An Example. For purposes of illustration, an example is included below. This example is fairly self-explanatory. The reader can see how the two underlying subgroups of the 10 x 10 matrix in Fig. 7 were found by using the technique that has been described above. Figure 8 shows the results of processing the matrix in Fig. 7.

Computer and Language. GROUPER is written in FORTRAN IV. On an IBM-360/65, a 100 x 100 matrix requires 70K. The total execution time for a complex 75 x 75 matrix with 100 iterations was 18.59 sec.

Availability. Documentation and a program listing may be obtained for $2 from the International Relations Research Institute, Department of International Relations, University of Southern California, Los Angeles, California 90007. Program source decks may be obtained for an additional $13.

REFERENCES

Beum, C. O., & Brundage, E. A method for analyzing the socio-matrix. Sociometry, 1950, 13, 141-145.

Sherwin, R. G. Interaction sub-groups in the international system. Paper presented at the meeting of the International Studies Association—West, Portland, Oregon, March 1972.