The rules of SPSS' Hierarchical Cluster Analysis for processing ties

Alexander M.J. Spaans
Willem A. van der Kloot
Department of Psychology
Leiden University
The Netherlands

November 1, 2005

In SPSS, hierarchical agglomerative clustering analysis of a similarity matrix uses the so-called Stored Matrix Approach[1]. Given an n x n proximity matrix **M** that contains the distances between n objects (i.e. clusters of one element), the algorithms[1] of the various methods consist of the following steps.

Step 1. Initialization.

Construct a table **U** of size n x n that (in the corresponding cells) only contains the elements that are below the main diagonal of **M**. For each row i (i = 2,…, n) in this "underdiagonal" Table**U,** determine the minimum distance value u_{ij} and store this value in Row i of a vector **v** of size n x 1. Store its corresponding column number j in Row i of a vector **c** (equally of size n x 1). Construct a vector **n** (of size n x 1) with the value 1 in each cell (i.e. the number of objects in each cluster).

Tie Rule 1

If, in a particular row k, the minimum value occurs more than once (i.e. in case of a tied minimum distance), choose the value with largest j as the minimum value and enter u_{kj} in row k of **v** and j in row k of **c**.

Step 2. Stage k (k = 1, …, n – 1) of the clustering process.

a. Determine row p with the minimum value, v_p, of **v** and find its corresponding column index, q, in **c** ($q = c_p$).

Tie Rule 2

In case of ties, choose the v_p with the largest p as the minimum value.

Thus, the most similar clusters to be merged are p and q.

[1] Anderberg, M.R. (1973). *Cluster analysis for applications*. New York: Academic Press.

b. Compute the similarities between this new cluster $\{p, q\}$ and the other clusters according to the rules of the cluster method chosen. For instance, in the Between-group average method (Baverage; also known as UPGMA), the distance between Cluster i and Cluster $\{p, q\}$ becomes

$$u_{i\{p,q\}} = (n_i n_p u_{ip} + n_i n_q u_{iq})/(n_i n_p + n_i n_q).$$

where n_i, n_p and n_q denote the numbers of objects in Clusters i, p, and q, respectively.

Label the new Cluster $\{p, q\}$ as q and insert its distances to the other clusters into Row q and Column q of **U.** Discard the elements in Row p and Column p of **U**. Update **n** by adding the value n_p to the value in Row q (i.e. $n_q = n_q + n_p$). Update **v** and **c** as follows:

i) For each row r in **U** for which $r = q$, or $c_r = p$, or $c_r = q$, determine and store the new minimum value and its corresponding column j, respectively, as v_r and c_r in **v** and **c**.

> **Tie Rule 3**
> In case of ties, choose the value with smallest j.

ii) For each of the updated similarities u_{rt} in **U** that are not in row q ($r \neq q$), and not in any row for which $c_r = p$ or $c_r = q$, update v_r and c_r if $u_{rt} < v_r$.

> **Tie Rule 4**
> In case of a tie, that is, if $u_{rt} = v_r$, the current minimum value v_r and corresponding column c_r remain unchanged.

Discard the elements of row p in **v**, **c**, and in **n**.

Now go back to Step 2, and let $k = k + 1$. Continue this cycle as long as $k < n$.

When $k = n - 1$, there are only two clusters left that can be merged; **U**, **v** and **c** then contain only one entry.

The handling of ties occur at Steps *1* (Rule 1)*, 2.a* (Rule 2), *2.c.i* (Rule 3) and *2.c.ii* (Rule 4). These rules are implicitly implemented in Anderberg's program source[1], that is, they were not explicitly mentioned or justified. The effect of these rules is that the breaking of ties depends on the stage of the agglomeration process. For instance, if minimum distance ties occur in the same row, three situations can be distinguished:

- the tied distances occur at the initialization step; in this case Rule 1 applies, that is, the distance with the *largest* column index is chosen.
- one of the tied minimum distance values was already represented in **v** and **c**; in this case Rule 4 applies, that is, the 'old' minimum is preferred, *regardless of its column position*.
- the minimum distance value was not already present in **v** but results from a new comparison of the row entries. In this case Rule 3 applies, that is, the distance with the *smallest* column index is preferred.

Together, these rules entail a rather complex dependency of any hierarchical clustering schedule on the order in which the data are read into the computer.

[1] Anderberg, M.R. (1973). *Cluster analysis for applications*. New York: Academic Press.

U contains the elements below the main diagonal of the matrix **M** of distances among six objects *A, B, C, D, E, F*.

U =

	A	*B*	*C*	*D*	*E*	*F*
A						
B	25					
C	17	19				
D	115	90	101			
E	102	76	89	22		
F	64	38	48	51	49	

Construct vector **v** where v_r = minimum of row $\mathbf{u}_r$ and vector **c** where c_r = the corresponding column index of the minimum of $\mathbf{u}_r$. Construct vector **n** with the number of objects in each cluster.

v =

25
17
90
22
38

c =

1
1
2
4
2

n =

1
1
1
1
1
1

Step $k = 1$

Find the minimum value in **v**: this is 17 in Row 3 with corresponding *c*-value of 1. Therefore, $p = 3$ and $q = 1$. Merge the clusters of Row *p* and Column *q*, that is, merge Objects *C* and *A*. Compute the distances between Cluster {*A, C*} and each of the remaining objects. Enter those distances in Row *q* and Column *q* of **U** (if $q = 1$ the rows of **U** remain unchanged). Delete the elements in Row *p* and in Column *p* from **U**. This yields

U =

	A,C	*B*		*D*	*E*	*F*
A,C						
B	22					
D	108	90				
E	95.5	76		22		
F	56	38		51	49	

[N.B.: the updated elements of **U** are highlighted]

Update **v**, **c**, and **n** as follows:

- add the value n_p to n_q (i.e. $n_1 = 1 + 1 = 2$).
- for Row $r = q$ of **U** determine the minimum value. Since $q = 1$, there are no elements in this row of **U**; therefore, nothing happens.
- for each row r in **U** for which $c_r = p$ (here: $p = 3$) determine the minimum value. Since there is no v_r with $c_r = p$, nothing happens.
- for each row r in **U** for which $c_r = q$ (here: $q = 1$) determine the minimum value; replace v_r by this minimum and c_r by the corresponding column index. As the third row of **U** was discarded, this applies only to the second row in **U**. The minimum in Row 2 of **U** is 22 in Column 2, thus the updated $v_2 = 22$ and the updated $c_2 = 1$.
- for all rows r of **U**, except those for which $r = q$, or $c_r = q$, or $c_r = p$, check whether the updated value(s) in those rows are smaller than the current value of v_r; if they are, update v_r and c_r. Thus, we need to check whether the updated values of u_{41}, u_{51}, and u_{61} are smaller than v_4, v_5, and v_6. As they are not, no further updates are necessary.
- discard row p from **v**, **c** and **n**.

This yields:

v =

22
90
22
38

c =

1
2
4
2

n =

2
1
1
1
1

Step $k = 2$

Find the minimum value in **v**. There are two minimum values (22): in Row 2 with corresponding $c_2 = 1$ and in Row 5 with $c_5 = 4$. **Apply Rule 2**: choose the distance with largest row-index p. Therefore, $p = 5$ and $q = 4$. Merge the clusters of Row p and Column q of **U**, that is, merge Clusters E and D.

Compute the distances between Cluster {*D, E*} and the remaining clusters. Enter those distances in row *q* and column *q* of **U**. Delete the elements in row *p* and column *p* from **U**.

U =

	A,C	*B*		*D,E*		*F*
A, C						
B	22					
D,E	101.75	83				
F	56	38		50		

Update **v**, **c**, and **n** as follows:

- augment Row *q* of **n** by n_p (i.e. $n_4 = 1 + 1 = 2$).
- for Row $r = q$ (here, $r = q = 4$) of **U** determine the minimum value; the minimum in Row 4 of U is 83. Replace v_4 by this minimum and c_4 by the corresponding column index (here: $c_4 = 2$).
- for each row *r* in **U** for which $c_r = p$ (here: $p = 5$) determine the minimum value. There is no such row in **U**, so nothing happens.
- for each row r in **U** for which $c_r = q$ (here: $q = 4$) determine the minimum value. There is no such row in **U**, so nothing happens.
- for all rows *r* of **U**, except those for which $r = q$, or $c_r = q$, or $c_r = p$, check whether the updated value(s) in those rows are smaller than the current value of v_r. Thus, we need to check whether the updated value of u_{64} is smaller than v_6. As this is not the case no further update is necessary.
- delete the elements of row *p* from **v**, **c** and **n**.

  This yields:

v =

22
83
38

c =

1
2
2

n =

2
1
2
1

Step $k = 3$

Find the minimum value in **v**: there is one minimum value (22) in Row 2 with corresponding $c_2 = 1$. Therefore, $p = 2$ and $q = 1$. Merge the clusters of Row p and Column q of **U**, that is, merge Clusters $\{A, C\}$ and B.

Compute the distances between Cluster $\{A, B, C\}$ and each of the remaining clusters.

Enter those distances in row q and column q of **U**. Delete the elements of row p and column p from **U**.

U =

	A,B,C			*D,E*		*F*
A,B,C						
D,E	95.5					
F	50			50		

Update **v**, **c**, and **n** as follows:

- augment Row q of **n** by n_p (i.e. $n_1 = 2 + 1 = 3$).
- for Row $r = q$ (here, $r = q = 1$) of **U** determine the minimum value. Row 1 of **U** is empty. Nothing changes.
- for each row r in **U** for which $c_r = p$ (here: $p = 2$) determine the minimum value. There are two such rows: Row 4 and Row 6. The minimum in Row 4 of **U** is 95.5 in Column 1, thus the updated $v_4 = 95.5$ and the updated $c_4 = 1$. Row 6 has two cells that contain the same minimum value 50 in Column 1 and Column 4. **Apply Rule 3**: choose the distance with the smallest column index. Therefore $v_6 = 50$ and $c_6 = 1$.
- for each row r in **U** for which $c_r = q$ (here: $q = 1$) determine the minimum value. There is no such row in **U**, so nothing happens.
- for all rows r of **U**, except those for which $r = q$, or $c_r = q$, or $c_r = p$, check whether the updated value(s) in those rows are smaller than the current value of v_r; if they are, update v_r and c_r There are no such rows left in **U**, so nothing happens.

- delete row p from **v**, **c**, and **n**.

This yields:

v =
95.5
50

c =
1
1

n =
3
2
1

Step $k = 4$

Find the minimum value in **v**: there is one minimum value (50) in Row 6 with corresponding $c_6 = 1$. Therefore, $p = 6$ and $q = 1$. Merge the clusters of Row p and Column q of **U**, that is, merge Clusters F and $\{A, B, C\}$. Compute the distance between Cluster $\{A, B, C, F\}$ and the remaining cluster $\{D, E\}$. Enter this distance in Row q and Column q of **U**. Delete the elements of Row p and Column p from **U**.

U =

	A,B,C,F			*D,E*		
A,B,C,F						
D,E	84.125					

Step $k = 5$

Merge Cluster $\{D, E\}$ and $\{A, B, C, F\}$. The agglomeration process is finished. The corresponding agglomeration schedule is:

Stage	Clusters merged		Fusion coefficient
1	*A*	*C*	17
2	*D*	*E*	22
3	*A,C*	*B*	22
4	*A,B,C*	*F*	50
5	*A,B,C,F*	*D,E*	84.125