

Displays for Statistics 5401

Lecture 40

December 12, 2005

Christopher Bingham, Instructor
612-625-1024

Class Web Page

<http://www.stat.umn.edu/~kb/classes/5401>

Copyright© Christopher Bingham 2005

Hierarchical Agglomerative Algorithm

- At the start (stage 0), the clusters are $\{1\}, \{2\}, \dots, \{N\}$, each object forming a cluster. Inter-cluster distances are $d_{\{i\}\{j\}} = d_{ij}$.
A cluster with one object is sometimes called a *singleton*.
- At each stage, distances d_{UV} between each pair U and V of clusters are computed. Then the two closest clusters (smallest d_{UV}) are combined to form a larger cluster. This *reduces the number of clusters by 1*.
- At the end (stage $N-1$), there is just *one* cluster $\{1, 2, \dots, N\}$ consisting of all N objects.

2

Question:

How should you define $d_{UV} = d(U, V)$ for sets U and V where U and/or V are sets of objects, not individual objects?

Once you can define d_{UV} , all stages of the clustering process are completely specified except for how to deal with tied distances, if ties are possible.

Methods

The "flavors" of hierarchical agglomerative clustering differ only in how d_{UV} is defined.

All the standard methods share features:

- You can compute the $(N-k-1) \times (N-k-1)$ matrix $\mathbf{D}^{(k+1)} = [d_{UV}^{(k+1)}]$ for stage $k+1$ with $N-k-1$ clusters from $\mathbf{D}^{(k)}$, and for some methods, the cluster sizes N_U, N_V and N_W .
- Distances between non-merged clusters are unchanged.

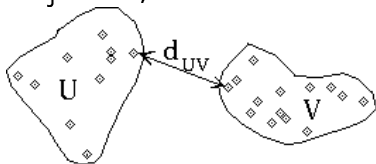
You need at most two (three with Ward, centroid, and median methods) rows and columns of $\mathbf{D}^{(k)}$ to compute a row and column of $\mathbf{D}^{(k+1)}$, plus possibly cluster sizes.

This is the "updating" ($\mathbf{D}^{(k)} \rightarrow \mathbf{D}^{(k+1)}$) step.

- **Single linkage** (minimum distance)

$$d_{UV} = d(U,V) \equiv \min_{i \in U, j \in V} d_{ij}$$

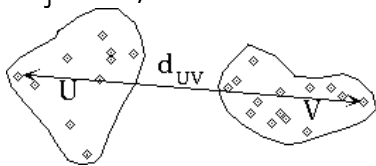
2 clusters are as close as the 2 closest objects, 1 from each cluster.



- **Complete linkage** (maximum distance)

$$d_{UV} \equiv d(U,V) = \max_{i \in U, j \in V} d_{ij}$$

2 clusters are as close as the 2 most *distant* objects, 1 from each cluster.



Geometrically motivated methods

Suppose you can define cluster "centers" C_U and C_V . Then when $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ are distances between data points, a measure of intercluster distance might be

$$d_{UV} = d(U,V) = d(C_U, C_V).$$

Or you might use cluster sizes N_U and N_V to adjust raw distances $d(C_U, C_V)$.

When distance updating rules don't use the centers explicitly you can use such methods even when "centers" make no sense, as is the case when all you have is a distance matrix D .

In my experience, complete linkage never seems to do a good job of clustering.

Single linkage sometimes does better, but tends to find long thin clusters.

- **Average linkage** (average distance)

$$d_{UV} = d(U,V) \equiv \sum_{i \in U, j \in V} d_{ij} / (N_U N_V)$$

This is an *average* distance since there are $N_U N_V$ pairs (i,j) , $i \in U, j \in V$.

This often works well and is a useful default.

- **Centroid method**

$d_{UV} = d(U,V) = d(C_U, C_V)$, where C_U is the "centroid" of the group.

When $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, C_U is $\bar{\mathbf{x}}_U$ and

$$d(U,V) = \|\bar{\mathbf{x}}_U - \bar{\mathbf{x}}_V\|^2 = \sum_{1 \leq k < p} (\bar{x}_{kU} - \bar{x}_{kV})^2$$

You need only the formula for $d_{w,\{U,V\}}^{(k+1)}$ so you can use the centroid method in the absence of any centroids.

You can use the same centers and updating formula for a generalized distance

$$d_{ij} = (\mathbf{x}_i - \mathbf{x}_j)' \mathbf{A}^{-1} (\mathbf{x}_i - \mathbf{x}_j),$$

where \mathbf{A} is positive definite.

• **Ward method**

This uses centroids as centers, but the distance between centers is weighted by a factor depending on cluster sizes.

$$d_{UV} = d(U,V) = (N_U N_V / (N_U + N_V)) \times d(C_U, C_V)$$

$$= d(C_U, C_V) / (1/N_U + 1/N_V)$$

When $d_{ij} = \|x_i - x_j\|^2$, $C_U = \bar{x}_U$,

$$d_{UV} = \text{trace } H_{UV}$$

where

$$H_{UV} \equiv (N_U N_V / (N_U + N_V)) (\bar{x}_U - \bar{x}_V) (\bar{x}_U - \bar{x}_V)'$$

is the *among groups* hypothesis matrix in a two group MANOVA.

Two large clusters are more distant than a pair of smaller clusters whose means are the same distance apart. This helps prevent merges of groups that are "significantly" different.

This works best with "spherical clusters" with $\Sigma_k \approx KI_p$.

Example from handout, N = 8, p = 2

Data were generated as samples from 3 bivariate normal populations.

	μ_i	Σ_i	n_i
π_1	$\begin{bmatrix} 10 \\ 30 \end{bmatrix}$	$\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$	3
π_2	$\begin{bmatrix} 15 \\ 28 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 2.25 \end{bmatrix}$	3
π_3	$\begin{bmatrix} 20 \\ 31 \end{bmatrix}$	$\begin{bmatrix} 2.25 & 0 \\ 0 & 0.25 \end{bmatrix}$	2

Here are the data

```
Cmd> x # row label is population number
      x1      x2
2     15.606  27.451
1     7.2295  29.53
1     9.9958  30.821
3     17.241  31.21
2     16.212  25.889
1     10.644  28.937
3     20.954  31.244
2     14.528  24.695
```

The row labels are population numbers which you would not have in a real problem.

• **Median method**

$d_{UV} = d(U,V) = d(C_U, C_V)$, but $C_{\{U,V\}}$ is half way between the C_U and C_V regardless of the size of the clusters.

• **McQuitty method**

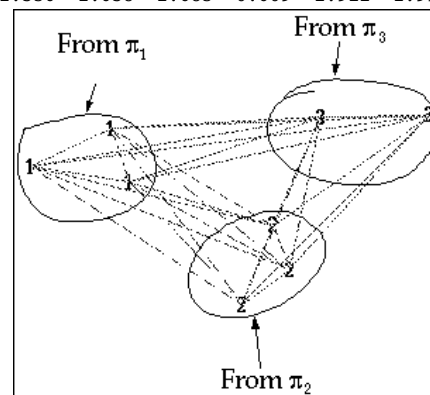
There is no clear geometry based on centers. It is defined solely in terms of an updating formula.

$$d_{w,\{U,V\}}^{(k+1)} = (d_{wU}^{(k)} + d_{wV}^{(k)}) / 2.$$

The distance from W to the new cluster {U,V} is the average of the distances from W to U and from W to V and makes no use of $d_{UV}^{(k)}$.

Compute the standardized distances between cases

```
Cmd> d <- matrix(dmat(8,0), \
  labels:structure(getlabels(x,1),getlabels(x,1)))
Cmd> xs <- standardize(x)
Cmd> for(i,1,7){for(j,i+1,8){ # loop over 1 <= i < j <= 8
  d[i,j] <- d[j,i] <- \
  sqrt(sum(vector((xs[i,]-xs[j,]))^2));}}
Cmd> print(d,format:"6.3f")
d:
      2      1      1      3      2      1      3      2
2  0.000  2.052  1.845  1.551  0.641  1.260  1.936  1.131
1  2.052  0.000  0.807  2.340  2.485  0.800  3.148  2.536
1  1.845  0.807  0.000  1.629  2.418  0.769  2.459  2.658
3  1.551  2.340  1.629  0.000  2.147  1.735  0.831  2.683
2  0.641  2.485  2.418  2.147  0.000  1.746  2.396  0.609
1  1.260  0.800  0.769  1.735  1.746  0.000  2.486  1.911
3  1.936  3.148  2.459  0.831  2.396  2.486  0.000  2.995
2  1.131  2.536  2.658  2.683  0.609  1.911  2.995  0.000
```



Plot with all lines between points.

Reorder distances so that groups are together.

```
Cmd> neworder <- vector(2,3,6,1,5,8,4,7)
Cmd> print(d[neworder, neworder],format:"6.3f")
MATRIX:
  1 1 1 2 2 2 3 3
1 | 0.000 0.807 0.800 2.052 2.485 2.536 2.340 3.148
1 | 0.807 0.000 0.769 1.845 2.418 2.658 1.629 2.459
1 | 0.800 0.769 0.000 1.260 1.746 1.911 1.735 2.486
2 | 2.052 1.845 1.260 0.000 0.641 1.131 1.551 1.936
2 | 2.485 2.418 1.746 0.641 0.000 0.609 2.147 2.396
2 | 2.536 2.658 1.911 1.131 0.609 0.000 2.683 2.995
3 | 2.340 1.629 1.735 1.551 2.147 2.683 0.000 0.831
3 | 3.148 2.459 2.486 1.936 2.396 2.995 0.831 0.000
```

Intra-cluster distances are smaller than inter-cluster distances.

That is the basis for most clustering algorithms.

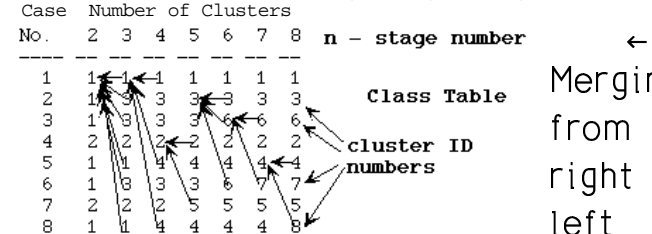
Here's a "magic" way to compute d:

```
Cmd> temp <- xs %**% xs'
Cmd> d <- sqrt(diag(temp) + diag(temp)' - 2*temp)
Cmd> print(d,format:"6.3f")
dd:
  2 1 1 3 2 1 3 2
2 | 0.000 2.052 1.845 1.551 0.641 1.260 1.936 1.131
1 | 2.052 0.000 0.807 2.340 2.485 0.800 3.148 2.536
1 | 1.845 0.807 0.000 1.629 2.418 0.769 2.459 2.658
3 | 1.551 2.340 1.629 0.000 2.147 1.735 0.831 2.683
2 | 0.641 2.485 2.418 2.147 0.000 1.746 2.396 0.609
1 | 1.260 0.800 0.769 1.735 1.746 0.000 2.486 1.911
3 | 1.936 3.148 2.459 0.831 2.396 2.486 0.000 2.995
2 | 1.131 2.536 2.658 2.683 0.609 1.911 2.995 0.000
```

Distances for standardized data.

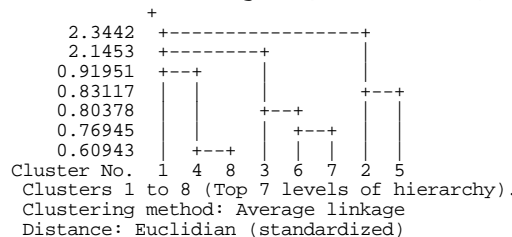
	2	1	1	3	2	1	3	2
2	0.000	2.052	1.845	1.551	<u>0.641</u>	1.260	1.936	1.131
1	2.052	0.000	0.807	2.340	2.485	0.800	3.148	2.536
1	1.845	0.807	0.000	1.629	2.418	<u>0.769</u>	2.459	2.658
3	1.551	2.340	1.629	0.000	2.147	1.735	0.831	2.683
2	<u>0.641</u>	2.485	2.418	2.147	0.000	1.746	2.396	<u>0.609</u>
1	1.260	0.800	<u>0.769</u>	1.735	1.746	0.000	2.486	1.911
3	1.936	3.148	2.459	0.831	2.396	2.486	0.000	2.995
2	1.131	2.536	2.658	2.683	<u>0.609</u>	1.911	2.995	0.000

```
Cmd> cluster(x) # cluster using Average Linkage (default)
```



Merging from right to left

Criterion Dendrogram (inverted tree)



Merging from bottom to top

Each step up is a joining of paths reflecting a merge.

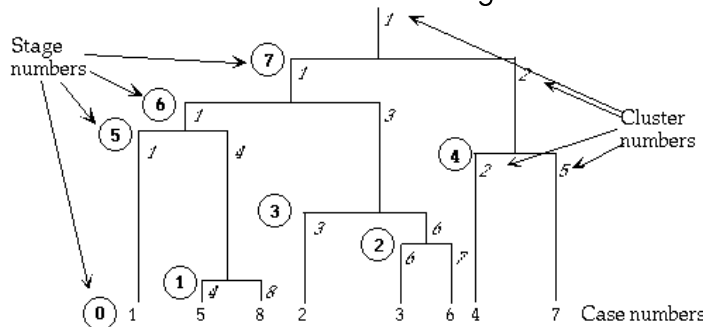
The numbers in the criterion column are the distances between the two clusters being merged at that stage.

A sharp increase going up the criterion column suggests the merge may have involved dissimilar clusters that perhaps should not be merged. Such jumps are the basis of common ways to choose the number of clusters.

Across the bottom are cluster (not case) numbers. The corresponding case numbers are (see the class table)

Cluster number	1	2	3	4	5	6	7	8
Case numbers	1	4	2	5	7	3	6	8

A more conventional dendrogram is



Cluster numbers are in italics, case numbers at bottom. Numbers in circles ④ are the stage numbers.

Often the vertical dimension in a dendrogram is the distance between the two clusters being merged (criterion in cluster() output). That way you can visually identify stages where there are sharp changes in the criterion.

A hard thing to understand in `cluster()` output is recognizing there is no simple matching of case numbers and cluster ID numbers.

You learn the matching *only* from the Class Table where each row goes with a case.

The numbers in the table are cluster numbers which `cluster()` assigns after clustering is complete. This is done in such a way that, at the stage when there are k clusters, cluster k is merged with a cluster with lower number.

- Stage 2 (merge clusters **6** and **7**)
 $1 = \{1\}$, $2 = \{4\}$, $3 = \{2\}$, $4 = \{5,8\}$, $5 = \{7\}$,
 $6 = \{3,6\}$

You merge $6 = \{3\}$ and $7 = \{6\}$ because all the new distances as well as $d_{1,4}$, $d_{1,2}$, etc. are larger than $d_{3,6} = .769$

- Stage 3 (merge clusters **3** and **6**)
 $1 = \{1\}$, $2 = \{4\}$, $3 = \{2,3,6\}$, $4 = \{5,8\}$, $5 = \{7\}$
- Stage 4 (merge clusters **2** and **5**)
- Stage 5 (merge clusters **1** and **4**)
 $1 = \{1,5,8\}$, $2 = \{4,7\}$, $3 = \{2,3,6\}$
- Stage 6 (merge clusters **1** and **3**)
 $1 = \{1,2,3,5,6,8\}$, $2 = \{4,7\}$
- Stage 7 $1 = \{1,2,3,4,5,6,7,8\}$ (merge clusters 1 and 2; not shown)

Details of average linkage process (**boldface** are cluster numbers; case numbers are in {...}):

- Stage 0 (all cases separate clusters)
 $1 = \{1\}$, $2 = \{4\}$, $3 = \{2\}$, $4 = \{5\}$, $5 = \{7\}$, $6 = \{3\}$,
 $7 = \{6\}$, $8 = \{8\}$
- Stage 1 (merge clusters **4** and **8**)
 $1 = \{1\}$, $2 = \{4\}$, $3 = \{2\}$, $4 = \{5,8\}$, $5 = \{7\}$,
 $6 = \{3\}$, $7 = \{6\}$

$4 = \{5\}$ and $8 = \{8\}$ were merged because minimum distance (0.609) is between cases 5 and 8

Update distances:

$$d(1,4) = d(\{1\},\{5,8\}) = (d_{1,5} + d_{1,8})/2 = (.641 + 1.131)/2 = 0.886,$$

$$d(2,4) = d(\{4\},\{5,8\}) = (d_{4,5} + d_{4,8})/2 = (2.147 + 2.683)/2 = 2.415,$$

$$d(3,4) = d(\{2\},\{5,8\}) = (d_{2,5} + d_{2,8})/2 = (2.485 + 2.536)/2 = 2.5105,$$

etc. All exceed .769

This history is summed up in this table

Clus ID	1	2	3	4	5	6	7	8
Stage 0	{1}	{4}	{2}	{5}	{7}	{3}	{6}	{8}
Stage 1	{1}	{4}	{2}	{5,8}	{7}	{3}	{6}	
Stage 2	{1}	{4}	{2}	{5,8}	{7}	{3,6}		
Stage 3	{1}	{4}	{2,3,6}	{5,8}	{7}			
Stage 4	{1}	{4,7}	{2,3,6}	{5,8}				
Stage 5	{1,5,8}	{4,7}	{2,3,6}					
Stage 6	{1,2,3,5,6,8}	{4,7}						
Stage 7	{1,2,3,4,5,6,7,8}							

From the jump in the criterion, the place to stop and accept the clusters found is at stage 5 when there are 3 clusters. In a real problem, it is never this clear.

Updating Formulas

- Single linkage

$$d_{w,\{U,V\}}^{(k+1)} = \min\{d_{wU}^{(k)}, d_{wV}^{(k)}\}$$

- Complete linkage

$$d_{w,\{U,V\}}^{(k+1)} = \max\{d_{wU}^{(k)}, d_{wV}^{(k)}\}$$

- Average linkage

$$d_{w,\{U,V\}}^{(k+1)} = (N_U d_{wU}^{(k)} + N_V d_{wV}^{(k)}) / (N_U + N_V)$$

This uses the sizes N_U and N_V of the clusters being merged, but not the size N_w of W or $d_{UV}^{(k)}$.

These "linkage" methods do not make use of $d_{UV}^{(k)}$ in updating.

21

- Median method

$$C_{\{U,V\}}^{(k+1)} = (C_U^{(k)} + C_V^{(k)}) / 2$$

$$d_{w,\{U,V\}}^{(k+1)} = (d_{wU}^{(k)} + d_{wV}^{(k)}) / 2 - d_{UV}^{(k)} / 4$$

The new center is half way between the old centers regardless of the size of the clusters.

- McQuitty method is defined purely by the updating formula

$$d_{w,\{U,V\}}^{(k+1)} = (d_{wU}^{(k)} + d_{wV}^{(k)}) / 2$$

23

- Centroid method

$$C_{\{U,V\}}^{(k+1)} = (N_U C_U^{(k)} + N_V C_V^{(k)}) / (N_U + N_V)$$

$$d_{w,\{U,V\}}^{(k+1)} = \frac{N_V d_{wU}^{(k)} + N_U d_{wV}^{(k)} - N_U N_V d_{UV}^{(k)}}{N_U + N_V}$$

Since only the formula for $d_{w,\{U,V\}}^{(k+1)}$ is needed, you can use the centroid method in the absence of real centroids.

- Ward method

$$C_{\{U,V\}}^{(k+1)} = (N_U C_U^{(k)} + N_V C_V^{(k)}) / (N_U + N_V)$$

$$d_{w,\{U,V\}}^{(k+1)} = \{M_{wU} d_{wU}^{(k)} / N_V + M_{wV} d_{wV}^{(k)} / N_U - d_{UV}^{(k)} / (N_U N_V)\}$$

with $M_{wU} = 1/N_U + 1/N_w$. This uses N_w , and is the only one of these methods that does so.

22

Usage of cluster()

`cluster(x, nclust:m, method:"average")`, x a n by p matrix,

- uses average linkage.
- prints the part of the class table describing 2 to m clusters
- prints the dendrogram for stages $N-m$, $N-m+1, \dots, N-1$ (m down to 1 cluster).

It does the full computation starting with N clusters but prints results only for stages with $\min(m, N)$ clusters or fewer.

Instead of "average" (the default) you can use "single", "complete", "ward", "mcquitty", "median" or "centroid" to specify other "flavors".

`cluster(dissim:d, ...)` and `cluster(simil:s, ...)` do cluster analysis on n by n dissimilarity or similarity matrices.

24

```
cluster(x,nclust:15,method:"average",\
reorder:T)
```

prints the class table with *reordered* rows, grouping cases into clusters. This makes it easier to see which cases are in each cluster.

```
Cmd> cluster(x,reorder:T,tree:F)
```

Case No.	Number of Clusters							
	2	3	4	5	6	7	8	
1	1	1	1	1	1	1	1	1
5	1	1	4	4	4	4	4	4
8	1	1	4	4	4	4	4	8
2	1	3	3	3	3	3	3	3
3	1	3	3	3	6	6	6	6
6	1	3	3	3	6	7	7	7
4	2	2	2	2	2	2	2	2
7	2	2	2	5	5	5	5	5

Cases are not in original order
 1st 3 from group 2, next 2 from group 1, last 3 from group 3

I suppressed printing the dendrogram by `tree:F`.

The original groups were

Group 1: {2, 3, 6} (cluster 3 of 3)

Group 2: {1, 5, 8} (cluster 1 of 3)

Group 3: {4, 7} (cluster 2 of 3)

The class table shows that at stage 5 the 3 clusters match these "true" clusters but have different numbers.

You can save the values of the criterion, the distances between objects, and the class table.

```
Cmd> stuff <- cluster(x,keep:"all") #prints nothing by default
```

```
Cmd> compnames(stuff)
```

```
(1) "distances"      8 by 8 symmetric matrix
(2) "classes"        class table
(3) "criterion"      distance apart of merged clusters
```

```
Cmd> print(format="6.3f", stuff)
```

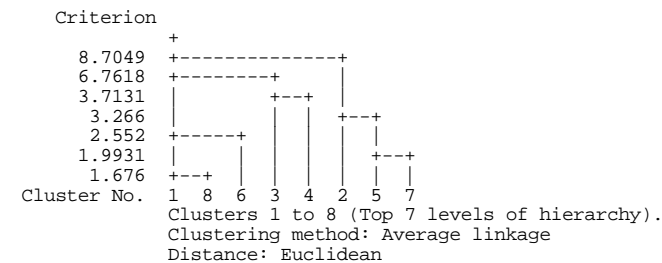
```
STRUCTURE:
component: distances      standardized distances
(1,1) 0.000 2.052 1.845 1.551 0.641 1.260 1.936 1.131
(2,1) 2.052 0.000 0.807 2.340 2.485 0.800 3.148 2.536
(3,1) 1.845 0.807 0.000 1.629 2.418 0.769 2.459 2.658
(4,1) 1.551 2.340 1.629 0.000 2.147 1.735 0.831 2.683
(5,1) 0.641 2.485 2.418 2.147 0.000 1.746 2.396 0.609
(6,1) 1.260 0.800 0.769 1.735 1.746 0.000 2.486 1.911
(7,1) 1.936 3.148 2.459 0.831 2.396 2.486 0.000 2.995
(8,1) 1.131 2.536 2.658 2.683 0.609 1.911 2.995 0.000
component: classes      Same as class table
(1,1) 1.000 1.000 1.000 1.000 1.000 1.000 1.000
(2,1) 1.000 3.000 3.000 3.000 3.000 3.000 3.000
(3,1) 1.000 3.000 3.000 3.000 6.000 6.000 6.000
(4,1) 2.000 2.000 2.000 2.000 2.000 2.000 2.000
(5,1) 1.000 1.000 4.000 4.000 4.000 4.000 4.000
(6,1) 1.000 3.000 3.000 3.000 6.000 7.000 7.000
(7,1) 2.000 2.000 2.000 5.000 5.000 5.000 5.000
(8,1) 1.000 1.000 4.000 4.000 4.000 4.000 8.000
component: criterion      in reverse order of computation
(1) 2.344 2.145 0.920 0.831 0.804 0.769 0.609
```

If you don't want everything, the value of `keep` can be "classes", "criterion", "distances", `vector("classes", "criterion")`, `vector("classes", "distances")` OR `vector("criterion", "distances")`

If you don't want to standardize before computing distances, use `standard:F`.

If you don't want to print the class table use `class:F`

```
Cmd> cluster(x,standard:F,class:F)
```



This printed only the dendrogram (`class:F`) and used *non-standardized* Euclidean distance (`standard:F`). The biggest jump in the criterion is at the stage when 3 clusters are merged to become 2 clusters.

The class table has $m-1$ columns, where m is the value of keyword `nclust` (default 9). The column labeled j summarizes the allocation into j clusters.

Using `keep` usually suppresses printing, but `tree:T` or `classes:T` force printing the dendrogram or the class table. `print:T` prints both.

Check one of the distances:

Macro `euclid(x,y)` computes Euclidean distance $\|x - y\|$ and `standardize(x)` standardizes the columns of x .

```
Cmd> euclid <- macro("sqrt(sum(vector((($1)-($2))^2))")
```

```
Cmd> xs <- standardize(x) # standardize data
```

```
Cmd> euclid(xs[1,],xs[2,])#row 1 - row 2 standardized distance
(1) 2.0519
```

Compare 2.0519 with the value in row 1, column 2 of the distance table above.