Classification Society of North America

Short Course[1]

A Combinatorial Introduction to Cluster Analysis

Melvin F. Janowitz

Associate Director
DIMACS (Center for Discrete Mathematics
and Theoretical Computer Science)
Rutgers, The State University
Piscataway, NJ 08854
email: melj@dimacs.rutgers.edu

June 13, 2002

[1]The contents of this tutorial represent the views of the author and not necessarily those of CSNA

Contents

1 What is Cluster Analysis?

This is a branch of exploratory data analysis that was (unfortunately for users) developed independently by people in a number of disciplines. This leads to some confusion in definitions and terminology. **Tip:** Be sure to check notation and definitions in any source you examine! Rather than try to give a formal definition of cluster analysis, we settle for describing a typical scenario.

Input Objects: $S = \{s_1, s_2, \ldots, s_n\}$, where $n \geq 3$.

Input Attributes: Collection $A = \{A_1, A_2, \ldots, A_t\}$ of attributes that the objects may have. Each A_i could be numerical (real numbers), ordinal (numbers having ordinal significance), nominal (numbers not involved), or binary (presence-absence with 0 for absent and 1 for present). Attributes taking values in a partially ordered set are best treated as nominal.

Object-attribute matrix: $M = m_{ij}$ is a square array having n rows and t columns. m_{ij} is the value of A_j for s_i.

Attributes need not all be the same type.

Goal: To find evidence of some inner structure for the data. Often the structure must be inferred solely on the basis of the attribute data. This might involve forming a *classification* of S. This is just a set $T_1, T_2, \ldots, T_k$ of subsets of S such that $S = \bigcup_i T_i$, and the T_i are not pairwise comparable. If the T_i are pairwise disjoint (as is almost always the case), we have a *partition*.

Example 1.1 $S = \{a, b, c, d, e, f, g\}$ with $T_1 = \{a, e, g\}$, $T_2 = \{a, b, c, d\}$, $T_3 = \{a, b, c, e\}$, and $T_4 = \{d, f, g\}$ is a classification, while $C_1 = \{a, b, d\}$. $C_2 = \{c, e, g\}$, $C_3 = \{f\}$ is a partition.

Other possibilities might involve forming a nested collection of partitions of the data, or representing the data pictorially. Often proceed through the intermediate step of first defining a numerical measure of similarity or dissimilarity. There are other types of inputs.

Illustrations of cluster algorithms will be based on the CLUSTAN cluster analysis package written by David Wishart.

2 Similarities and Dissimilarities

No standard terminology or notation. We will provide a commonly used setting. Different disciplines have their own terminology.

The Input:

Objects $S = \{s_1, s_2, \ldots, s_n\}$.

Attributes $A = \{A_1, A_2, \ldots, A_t\}$.

Object-Attribute Matrix $M = m_{ij}$.

2a Basic Definitions:

Similarity Measure or *Proximity Coefficient*:
Mapping $p : S \times S \mapsto \Re$ such that

$p(s_i, s_j) \geq 0$.

$p(s_i, s_j) = p(s_j, s_i)$.

$p(s_i, s_j) \leq p(s_h, s_h)$ for all i, j, h.

Idea: The higher the value of p, the more similar the pair of objects, so the highest value is $p(s_h, s_h)$. Evidently, $p(s_h, s_h) = p(s_k, s_k)$ for all h, k.

Dissimilarity Measure or *Dissimilarity Coefficient* (DC):
Mapping $d : S \times S \mapsto \Re$ such that

$d(s_i, s_i) = 0$.

$d(s_i, s_j) \geq 0$.

$d(s_i, s_j) = d(s_j, s_i)$.

Idea: Now higher values of d lead to less similar (i.e., more dissimilar) pairs of objects. Will work primarily with DCs.

Notation: People often write d_{ij} in place of $d(s_i, s_j)$.

Definition 2.1 The DC d is called

even if $d(s_i, s_j) = 0 \Longrightarrow d(s_i, s_k) = d(s_j, s_k)$ for all i, j, k;

definite if $d(s_i, s_j) = 0$ only when $s_i = s_j$;

a *metric* if it is definite and satisfies the *triangle inequality*

$$d(s_i, s_j) \leq d(s_i, s_k) + d(s_j, s_k) \text{ for all } i, j, k.$$

Many of the commonly used DCs are both even and definite.

Remark 2.2 Correspondence between similarities and dissimilarities:

If p is a similarity measure and d is defined by

$$d(s_i, s_j) = p(s_k, s_k) - p(s_i, s_j),$$

then d is a dissimilarity coefficient.

If d is a dissimilarity coefficient, if κ is a real number that is an upper bound for d, and if p is defined by

$$p(s_i, s_j) = \kappa - d(s_i, s_j),$$

then p is a similarity measure.

We make no attempt to present an exhaustive list of examples. Rather we shall just present some commonly used coefficients. Further information can be found in any of the standard tests on cluster analysis, as well as in [12]. The latest text on the market is [9]. It is a thin book that has many examples, and seems quite clear. Other useful recent books include [11] and [14]. Other references are mentioned at the end of the notes.

2b Numerical Data:

Let's try to keep the notation as simple as possible. We just have to indicate how to compute the dissimilarity or similarity between objects s_i and s_j. The attributes associated with any object can be thought of as a vector. So we are just dealing with two vectors. We agree to let

s_i correspond to the attributes $(m_1, m_2, \ldots, m_t)$

s_j correspond to the attributes $(n_1, n_2, \ldots, n_t)$

Squared Euclidean Distance

$$(m_1 - n_1)^2 + (m_2 - n_2)^2 + \cdots + (m_t - n_t)^2$$

Euclidean Distance

$$\sqrt{(m_1 - n_1)^2 + (m_2 - n_2)^2 + \cdots + (m_t - n_t)^2}$$

Manhattan Distance

$$\mid m_1 - n_1 \mid + \mid m_2 - n_2 \mid + \cdots + \mid m_t - n_t \mid$$

Sup Distance

$$\max \{ \mid m_1 - n_1 \mid, \mid m_2 - n_2 \mid, \ldots, \mid m_t - n_t \mid \}$$

Minkowski p-Metric, where $p \geq 1$ is fixed.

$$(\mid m_1 - n_1 \mid^p + \mid m_2 - n_2 \mid^p + \cdots + \mid m_t - n_t \mid^p)^{1/p}$$

Canberra Metric

$$\frac{\mid m_1 - n_1 \mid}{\mid m_1 \mid + \mid n_1 \mid} + \frac{\mid m_2 - n_2 \mid}{\mid m_2 \mid + \mid n_2 \mid} + \cdots + \frac{\mid m_t - n_t \mid}{\mid m_t \mid + \mid n_t \mid}$$

where one takes the ith term to be 0 if $m_i = n_i = 0$.

Bray-Curtis Coefficient

$$\frac{\mid m_1 - n_1 \mid + \mid m_2 - n_2 \mid + \cdots + \mid m_t - n_t \mid}{\mid m_1 + n_1 \mid + \mid m_2 + n_2 \mid + \cdots + \mid m_t + n_t \mid}$$

Correlation Coefficient This is one time when summation notation may make things easier to see. Let $\overline{m} = \frac{m_1+m_2+\cdots+m_t}{t}$, and similarly for $\overline{n}$. Then we take as the correlation between s_i and s_j

$$1 - \left| \frac{\sum_k (m_k - \overline{m})(n_k - \overline{n})}{\sqrt{\sum_k (m_k - \overline{m})^2 \sum_k (n_k - \overline{n})^2}} \right|$$

Note that no distinction is made here between positive and negative correlations. Not all authors agree.

We mention that many programs allow for weighting of the input attributes.

The choice of an appropriate dissimilarity measure depends largely on the nature of the data. A safe default is squared Euclidean distance. Apart from everything else there are some cluster algorithms that make little sense with other choices of distance. Generally, Manhattan distance and Euclidean distance are workable alternatives. If the input attributes have vastly different scales, then they can either first be normalized in some way, or correlation can be used as a dissimilarity, **Warning:** Normalization does not always lead to a better clustering.

An example from ecology: This is discussed in [20], p. 278. Suppose we are comparing abundance of three species at three sites.

	Species 1	Species 2	Species 3
Site s_1	0	1	1
Site s_2	1	0	0
Site s_3	0	4	8

Here are the dissimilarities produced by squared Euclidean distance, Manhattan distance Canberra distance, and the Bray-Curtis coefficient.

	$d(s_1, s_2)$	$d(s_1, s_3)$	$d(s_2, s_3)$
Squared Euclidean	3	58	81
Manhattan	3	10	13
Canberra	3	1.378	3
Bray-Curtis	1	.7143	1

This is called the *Species Abundance Paradox* in [20]. The paradox arises because under both Manhattan and squared Euclidean distance, s_1 and s_2 are the most similar – even though they share no species, while s_2s_3 is the least similar pair, though sharing two species. The Canberra and Bray-Curtis coefficients on the other hand seem better suited for this particular example. Thus Manhattan and squared Euclidean distance might not be appropriate for this particular data set. Of course if one is not interested in frequency counts – only in whether species are present or absent, the entry for site 3 might be 0 1 1 or 0 1/2 1, and this would present a different picture.

Do the various dissimilarity measures produce the same rankings? Most assuredly not. We illustrate this with a randomly generated attribute-object table. There are 4 objects and 5 attributes.

	A_1	A_2	A_3	A_4	A_5
s_1	13	3	12	20	8
s_2	17	20	6	6	9
s_3	17	18	5	16	14
s_4	10	6	14	3	5

We leave to the reader to verify the values of the indicated dissimilarity measures.

	s_1s_2	s_1s_3	s_1s_4	s_2s_3	s_2s_4	s_3s_4
Canberra	1.803	1.643	1.511	0.816	1.817	2.391
Manhattan	42	36	28	18	36	50
Sup	17	15	17	10	14	13
Squared Euclidean	538	342	320	130	334	524
Minkowski $p = 3$	19.9	16	17.1	10.4	15.5	17.9
Correlation	0.362	0.871	0.913	0.348	0.967	0.284
Bray-Curtis	0.368	0.286	0.298	0.141	0.375	0.463

Here are the rankings of the values for each dissimilarity coefficient.

	s_1s_2	s_1s_3	s_1s_4	s_2s_3	s_2s_4	s_3s_4
Canberra	4	3	2	1	5	6
Manhattan	4	3	2	1	3	5
Sup	5	4	5	1	3	2
Squared Euclidean	6	4	2	1	3	5
Minkowski $p = 3$	6	3	4	1	2	5
Correlation	3	4	5	2	6	1
Bray-Curtis	4	2	3	1	5	6

The message is clear. No mathematical model can tell you which dissimilarity measure to use. It is essential for the investigator to decide this, based on the nature of the data.

2c Binary Data

Binary data is presence/absence data. Use 0 to indicate absence and 1 to indicate presence. As in the numerical case, the literature abounds with dissimilarity coefficients suitable for use with binary data. We refer the reader to the standard texts as well as to [12]. We will content ourselves with just introducing and discussing a few of the more commonly used coefficients. The big decision you must make is whether you want to get any information from the shared absence of an attribute, an the effect of replacing an attribute by its negation. We will content ourselves with just introducing eight commonly used dissimilarity measures.

Here is the setup. Fix a pair of objects s_i and s_j. Suppose all attributes are binary. Let

s_i correspond to $(m_1, m_2, \ldots, m_t)$
s_j correspond to $(n_1, n_2, \ldots, n_t)$

Define a, b, c, d as indicated below.

$$\begin{aligned} a &= \text{count of } k \text{ such that } m_k = n_k = 1. \\ b &= \text{count of } k \text{ for which } m_k = 1 \text{ and } n_k = 0. \\ c &= \text{count of } k \text{ for which } m_k = 0 \text{ and } n_k = 1. \\ d &= \text{count of } k \text{ such that } m_k = n_k = 0. \\ n &= \text{total number of objects.} \end{aligned}$$

	Name	Dissimilarity	Similarity
(B1)	Jaccard	$\frac{b+c}{a+b+c}$	$\frac{a}{a+b+c}$
(B2)	Simple Matching	$\frac{b+c}{n}$	$\frac{a+d}{n}$
(B3)	Russel and Rao	$\frac{b+c+d}{n}$	$\frac{a}{n}$
(B4)	Sokal and Sneath #2	$\frac{2(b+c)}{a+2(b+c)}$	$\frac{a}{a+2(b+c)}$
(B5)	Rogers and Tanimoto	$\frac{2(b+c)}{a+2(b+c)+d}$	$\frac{a+d}{a+2(b+c)+d}$
(B6)	Dice	$\frac{b+c}{2a+b+c}$	$\frac{2a}{2a+b+c}$
(B7)	Sokal and Sneath #1	$\frac{b+c}{2a+b+c+2d}$	$\frac{2a+2d}{2a+b+c+2d}$
(B8)	Yule	$\frac{2bc}{ad+bc}$	$\frac{ad-bc}{ad+bc}$

Note that (B4) and (B7) are defined but not named in [26]. Note also how the presence or absence of d relates (B1),(B2) and (B3); (B4) and (B5); as well as (B6) and (B7). The Yule coefficient (Yule's Q coefficient) is motivated by the study of contingency tables. We shall not discuss its derivation. Let's play the same game we played with the numerical dissimilarities. We ask whether the above dissimilarities are saying the same thing – at least in an ordinal sense. We start by looking at some random binary attributes. We take 10 attributes on 5 objects as follows:

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
s_1	0	1	1	1	0	0	0	0	0	0
s_2	1	1	0	1	0	0	0	1	1	0
s_3	0	1	1	0	0	0	1	1	1	0
s_4	0	1	1	1	1	1	0	1	0	1
s_5	1	0	1	1	1	1	1	1	1	0

Here are the rankings of the various dissimilarity coefficients.

Object Pairs	12	13	14	15	23	24	25	34	35	45
Jaccard	4	4	3	5	3	4	2	4	2	1
Simple Matching	1	1	1	4	1	3	2	3	2	2
Russel and Rao	4	4	3	4	3	3	2	3	2	1
Sokal and Sneath #2	4	4	3	5	3	4	2	4	2	1
Rogers and Tanimoto	1	1	1	4	1	3	2	3	2	2
Dice	4	4	3	5	3	4	2	4	2	1
Sokal and Sneath #1	1	1	1	4	1	3	2	3	2	2
Yule	2	2	1	6	3	5	4	5	4	7

But notice how something strange has happened. The rankings for rows 1, 4 and 6 are identical, as are the rankings for rows 2, 5 and 7. Is this always going to be true? Well, here is where the mathematics comes in handy. It turns out that indeed this will always happen. The numbers may differ but the ranks remain the same. In his two papers [2] and [3], F. Baulieu studies binary dissimilarity measures. We just mention here his main result. He presents natural axioms that guarantee that his dissimilarity coefficients will all be of the form

$$d(s_i, s_j) = \frac{b+c}{\alpha a + b + c + \delta d},$$

where $\alpha > 0$ and $\delta \geq 0$. (Note: Of the binary DCs we introduced, only (B3) and (B8) are not of this form). He refers to such a dissimilarity coefficient by the symbol $D_{\alpha,\delta}$, and here is his main result.

Theorem 2.3 *$D_{\alpha,\delta}$, and $D_{\alpha',\delta'}$ induce the same output rankings if and only if $\alpha\delta' = \alpha'\delta$.*

Our observed equalities all follow from this. Indeed, the dissimilarities of the form $D_{\alpha,0}$ all produce the same rankings, as do those of the form $D_{\alpha,k\alpha}$, where $k > 0$. I do not know of similar results for other classes of dissimilarity coefficient. I mention though that for fixed $\theta > 0$, Gower and Legendre [12] consider metric and Euclidean properties of a family of binary dissimilarity measures of the form

$$D_\theta = \frac{\theta(b+c)}{a+d+\theta(b+c)} \text{ or } D'_\theta = \frac{\theta(b+c)}{a+\theta(b+c)}.$$

If we divide numerator and denominator by θ, we get $D_\theta = D_{\alpha,\alpha}$ and $D'_\theta = D_{\alpha,0}$, where $\alpha = 1/\theta$.

What follows is personal advice. You might find a different story in the literature. Again the choice of dissimilarity measure must depend on the nature of the data. If one wishes to ignore matched 0's, then Jaccard would seem a reasonable default choice, or possibly Russel and Rao. If matched 0's are to be counted then the simple matching coefficient would be a good default choice. These are all of the dissimilarities we plan to mention, but there are many others out there waiting to be chosen. As an example, we mention two dissimilarity formulas

$$\frac{(a+b)(a+c)-a^2}{(a+b)(a+c)} \text{ and } \frac{a^2d^2}{(a+b)(a+c)(b+d)(c+d)}$$

of a different type. We shall not mention either of these again. These and many other coefficients arise from statistical considerations that are based on other models.

2d Ordinal and Nominal Data

We view ordinal data as numerical where for some reason, one does not trust the actual values h_1, h_2, only whether $h_1 < h_2$. One way of dealing with ordinal data is to just rank order the attributes, possibly normalizing them so that the lowest rank is 0 and the highest is 1. Another method might involve the use of the Spearman or the Kendall rank order correlation coefficient. Still another method involves converting each ordinal attribute to a sequence of binary attributes. For example, if attribute A has three states a_1, a_2, a_3 with $a_1 < a_2 < a_3$, one can replace A with three binary attributes B_1, B_2, B_3, and recode the states of A as follows:

	B_1	B_2	B_3
s_1	1	0	0
s_2	1	1	0
s_3	1	1	1

Remark 2.4 If a dissimilarity coefficient d has only ordinal significance, the distinction implied by the triangle inequality disappears (see [11], p. 16). This fact is not often mentioned in the literature. It turns out that there are constants c_1, c_2 having the property that the DCs defined by $d(a,b) + c_1$ and $\sqrt{d(a,b)^2 + c_2}$ each satisfy the triangle inequality.

There are at least two ways to deal with nominal data. One can define $d(s_i, s_j) = k/t$, where k is the number of attributes in which s_i and s_j have different states, and t is the total number of attributes. Another would involve recoding the given attribute into a sequence of binary attributes. For example, if attribute A has three states red, green, or blue then one can just create three binary variables which together specify which state is present. Similar techniques can be used for attributes whose states form a partially ordered set.

There is no unanimous opinion about how to deal with mixed data. The issues are discussed in the standard references (e.g., see [11], p. 21, [17], pp. 32-37, [28], pp. 32-33). Here is a workable strategy. Divide the attributes into groups A_n, A_b, A_r, A_o according to whether they are nominal, binary, numerical or ordinal. Use an appropriate dissimilarity measure on each type of attribute to obtain dissimilarity measure d_n, d_b, d_r, d_o, and then define

$$d(s_i, s_j) = w_n d_n(s_i, s_j) + w_b d_b(s_i, s_j) + w_r d_r(s_i, s_j) + w_o d_o(s_i, s_j),$$

where w_n, w_b, w_r, w_o are appropriately chosen weight factors.

2e Standardization of attributes and missing values

When attributes are on quite different scales it is often desirable to standardize them in some way. Beware though that standardization is not always the right thing to do. Ultimately, you need to check to see if your output makes sense, and is even meaningful. David Wishart's Clustan program offers three techniques for standardization.

Standardize to z-scores: This converts each attribute to zero mean and unit variance. **Warning:** If the actual numbers are impor-

tant, you need to understand whether a given program is using the variance or the sample variance.

Standardize variable ranges: each attribute is scaled so its minimum value is 0, and its maximum value is 1.

Rescale to unit range: The entire object-attribute matrix is scaled so that its minimum value is 0 and its maximum value is 1.

At this point in the lecture. there does not seem to be much else to say on this issue. Equally, we do not have much to say about missing values. If you are using a commercial software package you need to discover how it handles missing values. It may estimate them, or omit any attribute having a missing value, or omit any value of a dissimilarity coefficient that involves a missing value. It may profit you to estimate on your own the needed missing values of attributes. At least you will understand what is happening.

3 Some clustering algorithms

We begin with some basic terminology. We are trying to determine the structure of a finite set S of objects. The building block is a *partition*. Recall that this is a collection P of nonempty subsets $C_1, C_2, \ldots, C_t$ of S whose union is S, and having the property that either $C_i = C_j$ or $C_i\,, C_j$ have no elements in common; i.e., the sets in a partition are *pairwise disjoint*. The sets making up a partition are called *clusters*. The partitions of S are partially ordered by the rule $P \leq Q$ if every cluster in Q is formed by merging two or more clusters from P. Clearly the smallest partition of S consists of the singleton subsets of S, and the largest one has a single class S. These will be denoted respectively as P_0 and P_S.

A *cluster method* usually produces either a single partition P, or a nested sequence of partitions $P_1 < P_2 < \cdots < P_k = P_S$. Often times the partitions are indexed by real numbers $h_1 < h_2 < \cdots < h_k$ to indicate the levels at which the P_i arise. These indices are sometimes

called *splitting levels* and sometimes called *fusion levels*. We examine first some of the methods that produce indexed sequences of partitions from a dissimilarity coefficient. So we have

attribute data $\mapsto$ dissimilarity coefficient $\mapsto$ partitions

or simply

dissimilarity coefficient $\mapsto$ partitions.

3a Agglomerative Algorithms

Efficient algorithms can be found in the literature (see [1], [6] or [7], for example). We outline instead an algorithm that is easy to visualize but not terribly efficient. It is an example of an *agglomerative cluster algorithm* in the sense that it starts with singletons, and successively merges clusters to form a nested sequence of partitions

$$P_1 < P_2 < \cdots < P_k = P_S.$$

1. INPUT dissimilarity coefficient d with image $0 = h_0 < h_1 < \cdots < h_u$. The initial clusters are singletons.

2. SET levnum = the lowest level at which a nontrivial pair of objects has a value.

3. Merge the first pair of objects you come to that have DC value levnum.

4. Change the data set to reflect the merger.

5. Use some method to determine the distance between the newly formed cluster and all others.

6. If a pair of clusters has distance levnum, go to Step 3.

7. Record the output clusters at this level. The dataset now becomes the existing set of clusters, and the DC is the revised DC on these clusters.

8. If all objects are clustered in a single cluster, exit.

9. Go to Step 2.

This will all best be illustrated with an example. The various methods we shall examine all relate to the various implementations of Step 5.

Example 3.1 We begin with a linear data set $S = \{s_1, s_2, s_3, s_4, s_5\}$, where $s_i = i$ for $1 = 1, 2, 3, 4, 5$. Our DC then is defined by $d(x, y) = |x - y|$. In tabular form, this produces

d	s_2	s_3	s_4	s_5
s_1	1	2	3	4
s_2		1	2	3
s_3			1	2
s_4				1

All right. This is admittedly an artificial example. One can argue that there is no structure to be discovered here. But let's see what happens. Single linkage clustering is a clustering method that merges clusters X, Y at the first level at which there is a link between them. Indeed, it proceeds by defining the dissimilarity between the merged cluster XY and cluster Z as $d(XY, Z) = \min\{d(X, Z), d(Y, Z)\}$. Single linkage clustering merges everything into a single cluster at level 1.

We turn next to a cluster method called *complete linkage* clustering. This is an agglomerative method that only merges clusters when all possible links have been made between them. This amounts to defining the dissimilarity between a merged cluster XY and cluster Z as $d(XY, Z) = \max\{d(X, Z), d(Y, Z)\}$. For the example at hand, we first merge s_1 and s_2 to form the new cluster s_1s_2. We now need to determine the dissimilarity between s_1s_2 and the remaining clusters. A typical calculation says that $d(s_1s_2, s_3) = \max\{d(s_1, s_3), d(s_2, s_3)\} = \max\{2, 1\} = 2$. This then produces

	s_3	s_4	s_5
s_1s_2	2	3	4
s_3		1	2
s_4			1

The next merger would be s_3 and s_4 to form s_3s_4. The updated dissimilarity is

	s_3s_4	s_5
s_1s_2	3	4
s_3s_4		2

All possible mergers have now been made at level 1, so the level 1 output is the partition whose classes are s_1s_2, s_3s_4, s_5. The next level is 2, so we form the cluster $s_3s_4s_5$ at level 2. The updated DC is

	$s_3s_4s_5$
s_1s_2	4

A final merger takes place at level 4.

There are at least four ways to represent the resulting output.

Dissimilarity coefficient: The idea here is to form the output DC by defining $d_{cl}(x, y)$ to be the lowest level at which x and y are merged.

d_{cl}	s_2	s_3	s_4	s_5
s_1	1	4	4	4
s_2		4	4	4
s_3			1	2
s_4				2

Clustan Version: Just indicate the cluster mergers and the levels at which they occur. The table indicates that $s1$ and $s2$ merge at level 1. Denote the new cluster as s_1. Then merge s_3 and s_4 at level 1 with the new cluster called s_3. Then merge s_3 and s_5 at level 2, calling the new cluster s_3. The final merger is at level 4, and merges s_1 and s_3.

First Cluster	Second Cluster	Fusion Value
s_1	s_2	1.000
s_3	s_4	1.000
s_3	s_5	2.000
s_1	s_3	4.000

List the partitions: At each level just list the non–singleton clusters.

Level	Nonsingular Clusters
1	$\{s_1, s_2\}$, $\{s_3, s_4\}$
2	$\{s_1, s_2\}$, $\{s_3, s_4, s_5\}$
4	$\{s_1, s_2, s_3, s_4, s_5\}$

Graphical Display: Either a vertical or horizontal tree can be used. Sometimes a scale is included to indicate the level at which each partition arises. This will not be done in these notes.

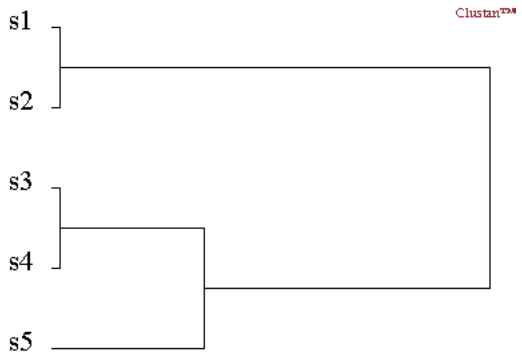

Figure 1: Complete Linkage Clustering on 5 Points

Single linkage and complete linkage clustering represent extremes. Single linkage tends to produce long stringy clusters, while complete linkage is such a harsh criterion that it tends to produce a few very compact clusters. Other techniques try to reach some sort of compromise between these criteria. One such technique is called *weighted average linkage* clustering (often abbreviated WPGMA in the literature). It follows the same format as complete linkage except that

the update for the distance between a merged cluster XY and cluster Z becomes the average of $d(X,Z)$ and $d(Y,Z)$. In other words, $d(XY,Z) = \frac{1}{2}\Big(d(X,Z) + d(Y,Z)\Big)$.

We illustrate WPGMA with the same example. The output for level 1 is identical with that of complete linkage, but the updated dissimilarity matrix changes.

	s_3s_4	s_5
s_1s_2	2	3.5
s_3s_4		1.5

As in complete linkage we now form the cluster $s_3s_4s_5$ at level 1.5, and calculate the upgraded matrix.

	$s_3s_4s_5$
s_1s_2	2.75

Here is the dissimilarity output for weighted average linkage clustering.

d_{wpgma}	$s2$	$s3$	$s4$	$s5$
$s1$	1	2.75	2.75	2.75
$s2$		2.75	2.75	2.75
$s3$			1	1.5
$s4$				1.5

Finally we have a technique called *unweighted pair group analysis* (abbreviated UPGMA). The update formula for $d(XY,Z)$ is given by

$$d(XY,Z) = \frac{|X|}{|X|+|Y|}d(X,Z) + \frac{|Y|}{|X|+|Y|}d(Y,Z).$$

It represents the average dissimilarity between an object in $X \cup Y$ and an object in Z. It may seem rather strange to call this an unweighted algorithm since there are clearly weighting constants involved. Indeed, some authors reverse the roles of UPGMA and WPGMA. The term "unweighted" refers to the role of the individual terms $d(x,z)$, $d(y,z)$ with $x \in X$, $y \in Y$ and $z \in Z$. The output from the UPGMA algorithm

is identical to WPGMA except that $d(s_1s_2, s_3s_4s_5)$ is 2.50 instead of 2.75.

The issue of ties: Before leaving this data set, let's say a word about ties. The DC for this data set shows dissimilarity 1 between the pairs $(s_1, s_2), (s_2, s_3), (s_3, s_4)$ and (s_4, s_5) and at level 1 forms the clusters s_1s_2 and s_3s_4, If the data were entered in the reverse order, then at level 1 we would form s_5s_4 and s_3s_2. Do we really want the cluster output to depend on the order in which the data is keyed into the computer? Most software developers do not see this as a problem. If you are using commercial software, you need to find out (if you can) how it handles tied data. At the very least you should experiment with reversing the order in which the data is entered to see if ties present a problem. The CLUSTAN package deals with this by telling you where there are ties and allowing you to reverse the order in which the ties are considered.

Jardine and Sibson [16] have a rather different solution to the problem. They just make all possible mergers at each level. This of course changes the clustering routine and the corresponding outputs. For example, the Jardine-Sibson version of complete linkage clustering would merge all the objects into a single cluster at level 1. Though your instructor likes their idea, the Jardine–Sibson technique has not caught on, so little further mention will be made of it. The interested reader can consult Figures 3, 4 and 5 for a more complicated example of the effect of ties.

Example 3.2 Here perhaps is a more interesting example. It is taken from [27]. Note though that we have only taken eight points from the sixteen considered in [27]. So the data consists of eight points in the Euclidean plane. We will change our format for referring to points and take as our set S the points $\{a, b, c, d, e, f, g, h\}$, and represent dissimilarities by δ rather than d. This is admittedly an artificial example, but it is nice to work with because of its natural geometric representation. To see the issues, try looking at Figure 2, and see if you can find a natural division into three clusters.

a	(0,4)	c	(1,5)	e	(3,3)	g	(2,1)
b	(0,3)	d	(2,4)	f	(2,2)	h	(1,0)

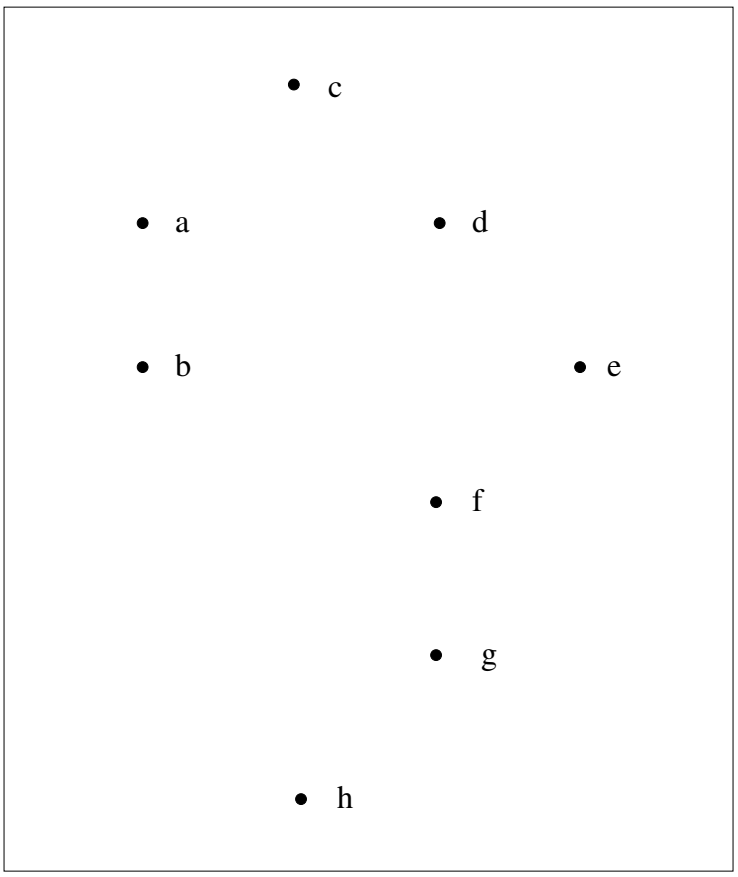

Figure 2: Eight point example

We present the DC formed using squared Euclidean distance, the intermediate DC formed at level 1 after the clusters ab and fg have formed, and finally the output DC. This is not terribly informative and is typical of the defects associated with single linkage clustering. There is a tendency toward forming long stringy clusters which group together objects that are only remotely related.

δ	b	c	d	e	f	g	h
a	1	2	4	10	8	13	17
b		5	5	9	5	8	10
c			2	8	10	17	25
d				2	4	9	17
e					2	5	13
f						1	5
g							2

Single Linkage:

	c	d	e	fg	h
ab	2	4	9	5	10
c		2	8	10	25
d			2	4	17
e				2	13
fg					2

$F_{sl}(\delta)$	b	c	d	e	f	g	h
a	1	2	2	2	2	2	2
b		2	2	2	2	2	2
c			2	2	2	2	2
d				2	2	2	2
e					2	2	2
f						1	2
g							2

Complete Linkage: We turn next to complete linkage clustering with the same example.

	c	d	e	fg	h
ab	5	5	10	13	17
c		2	8	17	25
d			2	9	17
e				5	13
fg					5

	cd	e	fg	h
ab	5	10	13	17
cd		8	17	25
e			5	13
fg				5

At level 5 the clusters $abcd$ and efg are formed. The updated dissimilarity matrix is

	efg	h
$abcd$	17	25
efg		13

Next, efg merges with h at level 13; the final merger takes place at level 25. Here is the output DC.

$F_{cl}(\delta)$	b	c	d	e	f	g	h
a	1	5	5	25	25	25	25
b		5	5	25	25	25	25
c			2	25	25	25	25
d				25	25	25	25
e					5	5	13
f						1	13
g							13

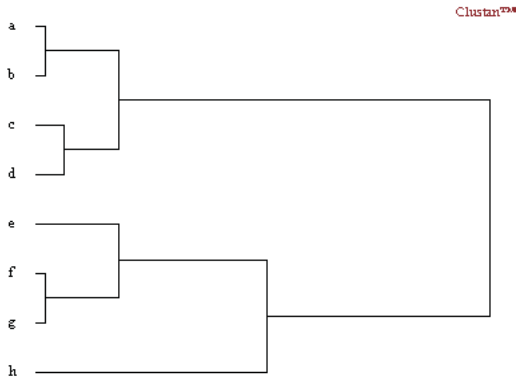

Figure 3: Complete Linkage Clustering on Eight Point Example

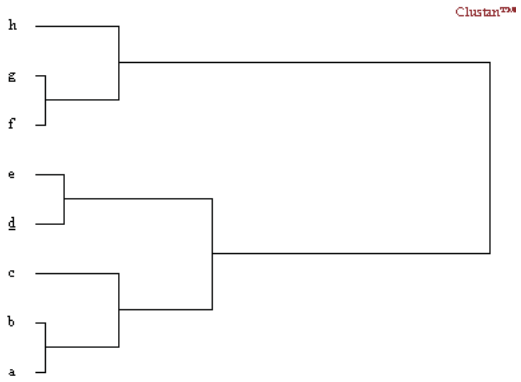

Figure 4: Complete Linkage Clustering with Order Reversed

Please compare Figures 3 and 4 to see how ties can dramatically affect a cluster output. The data sets were identical for both figures, squared Euclidean distance was the DC, and Complete Linkage the clustering algorithm. The only difference was the order in which the data was entered into the computer. Figure 5 illustrates the Jardine-Sibson method of dealing with ties. It may be illuminating to just compare the manner in which d and e are clustered in the Figures.

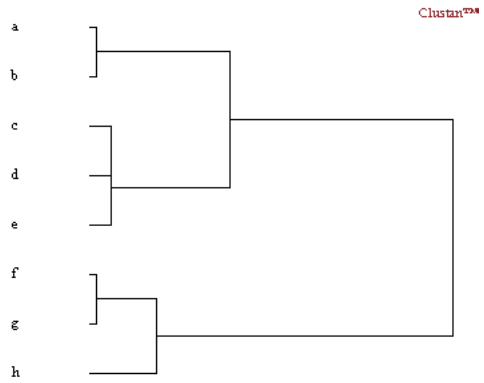

Figure 5: Jardine-Sibson Version of Complete Linkage Clustering

UPGMA: Here next is the UPGMA algorithm.

	c	d	e	fg	h
ab	3.5	4.5	9.5	8.5	13.5
c		2	8	13.5	25
d			2	6.5	17
e				3.5	13
fg					3.5

	cd	e	fg	h
ab	4	9.5	8.5	13.5
cd		5	10	21
e			3.5	13
fg				3.5

	cd	efg	h
ab	4	8.83	13.5
cd		8.33	21
efg			6.67

	efg	h
$abcd$	8.58	17.25
efg		6.67

The cluster $efgh$ is formed at level 6.67, and the final merger takes place at level 10.75. Here is the output DC.

$F_{upgma}(\delta)$	b	c	d	e	f	g	h
a	1	4	4	10.75	10.75	10.75	10.75
b		4	4	10.75	10.75	10.75	10.75
c			2	10.75	10.75	10.75	10.75
d				10.75	10.75	10.75	10.75
e					3.5	3.5	6.67
f						1	6.67
g							6.67

WPGMA: Now we turn to WPGMA. Up to and including level 2, the output is identical to that of UPGMA. At level 2, we have clusters ab, cd, e, fg, h and we form efg at level 3.5. We need to update the dissimilarity matrix.

	cd	efg	h
ab	4	9	13.5
cd		7.5	21
efg			8.25

	efg	h
$abcd$	8.25	17.25
efg		8.25

	h
$abcdefg$	12.75

Here now is the final DC.

$F_{upgma}(\delta)$	b	c	d	e	f	g	h
a	1	4	4	8.25	8.25	8.25	12.75
b		4	4	8.25	8.25	8.25	12.75
c			2	8.25	8.25	8.25	12.75
d				8.25	8.25	8.25	12.75
e					3.5	3.5	12.75
f						1	12.75
g							12.75

UPGMC and WPGMC: These cluster methods have a natural geometric interpretation if one thinks of the objects as being points in an n-dimensional Euclidean space. The distance between two clusters is taken to be the distance between their centroids. The WPGMC method upgrades the centroid of a merged cluster XY as the centroid of the two centroids for X and Y, whereas UPGMC computes the centroid of all $x \in X$ and that of all $y \in Y$. UPGMC is sometimes called centroid clustering, while WPGMC is called the median method. These can all profitably be regarded as special cases of a general clustering strategy called *flexible clustering* which is due to Lance and Williams [18]. This is essentially just a formula for $\delta(XY, Z)$ when cluster X and Y are merged. More general versions exist, but we shall discuss a version that involves constants $\alpha_{_X}$, $\alpha_{_Y}$ and β:

$$\delta(XY, Z) = \alpha_{_X}\delta(X, Z) + \alpha_{_Y}\delta(Y, Z) + \beta\delta(X, Y).$$

For WPGMC we take $\alpha_{_X} = \alpha_{_Y} = 1/2$ with $\beta = -1/4$. For UPGMC, we take

$$\alpha_{_X} = \frac{|X|}{|X| + |Y|},\ \alpha_{_Y} = \frac{|Y|}{|X| + |Y|},\ \beta = \frac{-|X||Y|}{(|X| + |Y|)^2}.$$

Warning: The geometric interpretation of these update formulas is only valid when used with squared Euclidean distance. A detailed discussion of how the formulas are derived can be found in [17], pp. 227–230. Despite the attractive geometric interpretation, they have a flaw called reversals. It sometimes happens that when clusters X, Y are merged, the distance $\delta(XY, Z) < \min\{\delta(X, Z), \delta(Y, Z)\}$. This makes the results a little hard to interpret. To see how this can happen, imagine a triangle ABC with vertices at $A(-.5, 0)$, $B(.5, 0)$, and $C(0, K)$, where $3/4 < K^2 < 1$. Using squared Euclidean distance, then $\delta(A, B) = 1$, and $\delta(A, C) = \delta(B, C) = K^2 + 1/4 > 1$. The cluster AB is formed and its centroid is at the origin. Then $\delta(AB, C) = K^2 < 1$ so we have a reversal. All of this is depicted in Figure 6, where D has coordinates (0,1) and E has coordinates $(0, \sqrt{3}/2)$, so triangle ABE (not depicted) is an equilateral triangle.

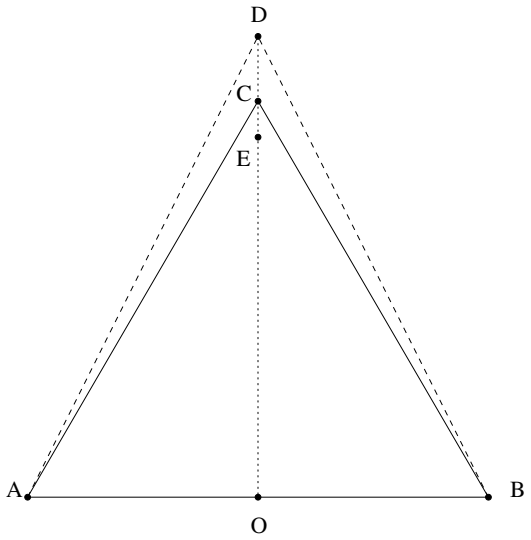

Figure 6: Example of a Reversal

Examples of these clustering methods may be found in Figures 12 and 13.

Ward's Method: (Incremental sum of squares) This method illustrates a class of agglomerative cluster methods that seek at each stage to optimize some external objective function. In this case it is the within cluster sum of squares of the pairwise distances. The basic assumption here is that the underlying objects are viewed as vectors in a suitable Euclidean vector space. These ideas have their motivation in the algorithms for Analysis of Variance. Though this is most certainly not how Ward [30] viewed the problems in his original work, it turns out that the algorithm may be viewed as a special case of flexible clustering with

$$\alpha_{_X} = \frac{|X| + |Z|}{|X| + |Y| + |Z|},\ \alpha_{_Y} = \frac{|Y| + |Z|}{|X| + |Y| + |Z|},\ \beta = \frac{-|Z|}{|X| + |Y| + |Z|}.$$

Any of the standard texts on cluster analysis should have a treatment of this method. One problem with algorithms of this type is that optimizing an external criterion at level h might not lead to optimality at a level beyond h.

Flexible Clustering We have mentioned flexible clustering several times. Most texts carry some sort of discussion of the subject. One such source is [11], p. 79. We summarize what we have mentioned. Suppose we are merging clusters X and Y, and want the dissimilarity between the new clusters XY and cluster Z. Letting $n_1 = |X|$, $n_2 = |Y|$, $n_3 = |Z|$, and $n = n_1 + n_2 + n_3$, we have

	$\alpha_i (i = 1, 2)$	β
WPGMA	$\frac{1}{2}$	0
UPGMA	$\frac{n_i}{n_1+n_2}$	0
WPGMC	$\frac{1}{2}$	$-\frac{1}{4}$
UPGMC	$\frac{n_i}{n_1+n_2}$	$\frac{-n_1 n_2}{(n_1+n_2)^2}$
Ward	$\frac{n_i+n_3}{n}$	$\frac{-n_3}{n}$

Recall here that $d(XY, Z) = \alpha_1 d(X, Z) + \alpha_2 d(Y, Z) + \beta d(X, Y)$

3b Divisive Algorithms

Divisive cluster algorithms proceed from the top down. They start with a single cluster and divide this into a number (often 2) of subclusters, and then successively divide the subclusters to form a nested sequence of partitions. Since one is often primarily interested in the higher level clusters one can argue that perhaps these techniques might be superior to the agglomerative algorithms. But like the agglomerative algorithms these are just stepwise optimal and there is no guarantee that they will be globally optimal. They also suffer from issues of computational complexity. The problem of looking at all possible divisions of an n element set into two classes can be shown to be NP-complete, so one is either limited to relatively small data sets, or one must use some heuristic to find "good" partitions. One such method might involve using a single attribute to make the division. This type of cluster method is called *monothetic* and is described in [11], pp. 130-134. The obvious problem is that clusters are being formed on the basis of a single attribute, while the desired clusters might be multivariate in nature.

Other techniques involve the optimization of certain objective functions. Typical functions for a cluster A include

diameter of A: $\max\{d(x,y) : x,y \in A\}$, and

within cluster sum of A: $\sum\{d(x,y) : x,y \in A\}$

Polynomial time algorithms exist to find a partition that minimizes certain objective functions for bipartitions. Among them are the maximum of the two cluster diameters and their sum. References are provided in [11]. We shall content ourselves with illustrating the general ideas with the data from Example 3.2. As in our earlier calculations, we use squared Euclidean distance as the DC. Since this is an 8 element set there are 127 partitions to consider. Indeed, for an n element set there are $2^{n-1} - 1$ distinct partitions with two classes. With such a small data set it is easy to do an exhaustive search., and this is precisely what we did. We used five criteria as described in the next table. The table entries represent the rank of the relevant statistic.

	$abcdefgh$	$abcdefgh$	$abcdefgh$	$acdefgbgh$	$acdebfgh$
sum of sums	9	2	1	6	5
max of sums	24	6	1	9	2
sum of diameters	1	2	4	6	6
max of diameters	1	1	2	1	1
within cluster DC	4	2	1	7	3

Note how the different criteria produce different optimal bipartitions. So there is a moral lesson here. Computer heuristics need not produce the best results.

3c Iterative relocation methods

We only discuss one illustrative method here: *k-means* clustering. The standard texts all treat this, and the commercial software packages should all have a version. We shall discuss some of what the Clustan program does. The idea is to produce a single partition having k classes. Any dissimilarity measure can be used, but the motivation for what is done assumes squared Euclidean distance. The program can either be viewed as the initial creation of k clusters or the refinement of a k cluster output that came from some other algorithm. The input is an n element data set with t numerical attributes and a dissimilarity measure.

Step 1. Choose k seed points. These are the starting points for the procedure. They can either be imported from some other algorithm, randomly chosen, arbitrarily chosen, or perhaps just taken to be the first k objects in the data set. They need not even be actual data points.

Step 2. Assign each object to the seed to which it is closest.

Step 3. Recompute the k cluster centers.

Step 4. Iterate the procedure until it either converges or the count exceeds some threshold. In Clustan the default is 10 iterations, though this can be reset.

It should be mentioned that there are ways to check for outliers, and these are placed in an unclassified residue. k-means has the advantage that it can be used on very large data sets; its disadvantage is the extent to which outputs are influenced by the initial choice of seed points.

We illustrate this with the data from Example 3.1. Recall that $s_i = i$ for $i = 1, 2, 3, 4, 5$ and use squared Euclidean distance as the DC. Let's take for initial seeds 1 and 2. This leads to the two clusters $A = \{1\}$ and $B = \{2, 3, 4, 5\}$. The cluster centers are 1 and 3.5, thus producing clusters $A = \{1, 2\}$ and $\{3, 4, 5\}$. At the next stage the cluster centers are 1.5 and 4, and stability has been achieved. Of course had we chosen 4 and 5 as the initial seeds, the output would have been $\{1, 2, 3\}$ and $\{4, 5\}$. When applied to the data from Example 3.2, the k-means algorithm produces the bipartition $\{a, b, c, d\}$, $\{e, f, g, h\}$.

4 A model for clustering algorithms

This material is based on ideas of Jardine and Sibson [16], though the presentation will look rather different from theirs. The goal here is to show how mathematics can help in understanding what it is that cluster algorithms are doing, and equally how it can prove misleading if it is not properly used. A reader uncomfortable with mathematical formalism may safely just omit this section. The first order of business is to identify the nature of the output of a clustering algorithm. First we need some mathematical machinery.

A *relation* on S is a set R of ordered pairs. We sometimes write xRy in place of $(x, y) \in R$. The relation R is

reflexive if xRx for all $x \in S$;

symmetric if xRy implies yRx;

transitive if xRy together with yRz implies xRz.

Relations that are reflexive, symmetric and transitive are called *equivalence relations*. Partitions are naturally associated with equivalence relations. For E an equivalence relation, the associated partition is

given by $\{[x] : x \in E\}$, where $[x] = \{y \in S : xEy\}$. $[x]$ is called the equivalence class determined by x.

Relations on S are partially ordered by the rule $R_1 \subseteq R_2$ if xR_1y implies xR_2y. The smallest equivalence relation is then $\Delta = \{(x, x) : x \in S\}$, and the largest is $S \times S$.

For a given DC d, and for any $h \geq 0$, we define $Td(h) = \{(a, b) : d(a, b) \leq h\}$. Each $Td(h)$ is a reflexive symmetric relation. The dissimilarity d is an *ultrametric* if every $Td(h)$ is an equivalence relation. This is clearly equivalent to the validity of the *ultrametric inequality*:

DCU $d(a, b) \leq \max\{d(a, c), d(b, c)\}$ for all $a, b, c \in E$.

If a clustering algorithm produces partitions at each level, and if we define $u(a, b)$ to be the first level at which a and b are first clustered, then u is an ultrametric (often called the cophenetic coefficient) from which the output can be recaptured. Of course we have been doing this all along, but in a more informal manner. Because of this, a cluster method may be viewed as a transformation of DCs into ultrametrics (or more generally into some other kind of DC). We list a few reasonable properties and then look at their consequences.

(A1) $F = F \circ F$. This says that F is idempotent.

(A2) If d' is obtained from d by relabelling E, then $F(d')$ should be obtained from $F(d)$ by that same relabelling.

(A3) If $d' = \alpha d$ $(\alpha > 0)$, then $F(d') = \alpha F(d)$.

(A3′) If $d' = \theta d$ for any monotone transformation θ of $[0, \infty)$, then $F(d') = \theta F(d)$.

(A4) F leaves every ultrametric fixed.

(A4′) The image of F is the set of ultrametrics.

(A5) F is isotone in that $d \leq d'$ implies that $F(d) \leq F(d')$.

(A6) If we use Euclidean distance as the metric on DCs, then F is continuous.

Though these axioms seem reasonable, when put together they seem much too strong.

(A1) This one is not at all obvious. The idea is that F has transformed a DC into a desired form. Once there, F should not alter the DC. But one can imagine algorithms that take several iterations before the desired output is reached.

(A2) Innocent looking but strong. The easiest implementation is at each level to merge all pairs that meet the clustering criterion.

(A3) Very innocent. If the scale of measurement is changed, this should not affect the output. Note that says nothing about changing to a logarithmic scale! It only involves linear scale changes.

(A3′) This is called *monotone equivariance* in [16], and represents a form of ordinal clustering.

(A4) This seems reasonable if we accept (A1).

(A4′) Many algorithms implicitly assume this.

(A5) This looks reasonable since $d \leq d'$ means that $d(a,b) \leq d'(a,b)$ for all $a, b \in E$. But it has dramatic consequences. Indeed, the consequences are so strong that one may very well question the validity of the axiom.

(A6) The idea here is that small errors in the input should not translate into large output errors. This axiom is subtle.

Theorem 4.1 *There is exactly one cluster algorithm that satisfies* (A1), (A4′), *and* (A5). *It is called single linkage clustering, and it is defined by taking for output at each level the transitive closure of* $Td(h)$. *The remaining axioms are all also satisfied.*

Theorem 4.2 *If* F *satisfies* (A1), (A4) *and* (A5), *and if* Z *is the image of* F, *then for any DC* d, $F(d) = \max\{u \in Z : u \leq d\}$.

Theorem 4.3 *If* F *satisfies* (A3) *and* (A5), *then* F *is continuous at all definite DCs. This is the strongest possible result.*

Theorem 4.4 *The following conditions are equivalent:*

(1) *F satisfies* (JS3′) *and is continuous.*

(2) *F satisfies* (JS3′) *and is right continuous.*

(3) *$F(\theta d) = \theta F(d)$ for every 0-preserving monotone mapping on $[0, \infty)$.*

(4) *There exists a mapping γ on the binary relations such that*

$$TF(d) = \gamma \circ Td.$$

If F satisfies (A3) and (A5), it is in fact true that F is left continuous in the sense that if $\lim_n d_n = d$ with $d_n \leq d$, then $\lim_n F(d_n) = F(d)$. Rather surprisingly, right continuity is equivalent to continuity. The connection with ordinal considerations can be motivated by the fact that for any DC d, there corresponds a positive constant ε such that whenever $\Delta(d, d') < \varepsilon$, then $d(a, b) < d(x, y) \Longrightarrow d'(a, b) < d'(x, y)$. Here $\Delta(d, d') = \sum\{(d(a, b) - d'(a, b))^2 : a, b \in E\}$. If we define $d \preceq d'$ to mean that $d(a, b) < d(x, y) \Longrightarrow d'(a, b) < d'(x, y)$, it seems reasonable to say that continuous cluster functions should have the property that $d \preceq d' \Longrightarrow F(d) \preceq F(d')$. But these are precisely the cluster functions that are suitable for use with ordinal data, since they have the property that if $d(a, b) < d(x, y)$ is equivalent to $d'(a, b) < d'(x, y)$, the same would be true for $F(d)$ and $F(d')$. It would seem that continuity is somewhat a secondary condition implied by other more basic ordinal considerations. A key desirable condition is that a cluster algorithm be stable under a few reversals of order. Such a reversal means that one has $d(a, b) < d(x, y)$, when in reality the input should have been $d(a, b) > d(x, y)$.

Much more can be said about continuity. Some details may be found in [5]. As a historical note, the Jardine-Sibson axioms only allowed single linkage clustering as the acceptable method. Continuity was the assumption people criticized. But continuity is implied by the remaining axioms, it is curious that a condition that involves numbers being close to each other should be implied by ordinal conditions.

Jardine and Sibson are not the only authors who have introduced properties that cluster functions may enjoy. Fisher and Van Ness [10] introduced admissibility conditions. They call a cluster function *monotone admissible* if applying a monotone mapping θ to d does not change any of the partitions formed by $F(d)$. There is an obvious connection to axiom (A3′). These and other issues are discussed in [11], pp. 98–100, where additional references may also be found. Other ideas may be found in [18].

5 Validation

A cluster algorithm will always produce clusters; How do we know that the clusters are not artifacts of the algorithm? This is not an easy question to answer. Related to this is the issue of when to stop. How many clusters are there? If a hierarchical tree is produced, what levels of the tree are significant? Most text books on clustering have some sort of discussion, but there is no definitive theory. Indeed, the literature has many lengthy discussions on the subject. Milligan [22] analyzes 30 different measures of this type by means of Monte Carlo methods. We just do not have the time to get into this, and there is no definitive theory. Indeed, though others may disagree, my opinion is that the techniques have little sound statistical basis, though they may be motivated by statistical considerations. For these reasons, we are just going to take a quick look at a few criteria. We will illustrate the criteria with data from Example 5.1. This example is from the Clustan distribution package and consists of an analysis of milk from 25 species of mammal. The cluster methods are single linkage SL, complete linkage CL, WPGMA , UPGMA, WPGMC, and UPGMC. This is for illustrative purposes only, and one should be cautious about drawing conclusions.

Gowers's statistic: This is just the sum of the squares of the differences between the input and the output DCs. It is a measure of the distance between the two DCs when they are considered to be vectors.

Cophenetic Correlation Coefficient (CPCC): This is highly recommended in [27]. The name is somewhat historical, but we are looking only at the ordinary product moment correlation between the input and output dissimilarities. Though it gives an indication of goodness of fit, it is difficult to calibrate because we are dealing with unknown distributions of the data.

The Goodman-Kruskal γ-Coefficient: Suppose the input DC is d and the output is u. Ideally, we want $d(a,b) < d(x,y)$ to imply failure of $u(a,b) > u(x,y)$. This coefficient measures the extent to which this is true. Let S be the set of all pairs ab, xy of pairs of objects such that $d(a,b) < d(x,y)$. The pair ab, xy is called *concordant* if $u(a,b) < u(x,y)$ and *discordant* if $u(a,b) > u(x,y)$. Ties are ignored. The statistic is then defined by

$$\gamma = \frac{S_+ - S_-}{S_+ + S_-},$$

where S_+ denotes the number of concordant pairs and S_- the number of discordant pairs. It can be used to compare the effectiveness of cluster methods operating on the same input data.

	SL	CL	WPGMA	UPGMA	WPGMC	UPGMC
Gower	1713	1635	566	633	2810	1059
Correlation	.69	.59	.76	.74	.76	.80
Gamma	.62	.72	.80	.79	.80	.80

Whatever else one may conclude, this shows that these three statistics give somewhat different answers. As we said earlier, they should only be used for guidance.

We turn now to the issue of the stopping point. How many clusters are there? The Clustan software has two stopping rules, both based on the papers [24, 25]. There is also a new algorithm under development. This we now describe.

The Clustan Algorithm: This is currently under development by David Wishart. Here is how one version of it works. We are

given a data matrix and use squared Euclidean distance as the DC. A cluster algorithm is applied. One wants to test the clusters for significance. Each column of the data matrix is randomly permuted and a new DC is computed. The fusion level for the output is computed corresponding to each number of clusters This is repeated a fixed number of times and a mean and standard deviation are computed for the fusion level corresponding to a k-cluster partition. This is compared with the output at hand. If the current output at level k is more than 1 standard deviation beneath the mean it is deemed significant. The randomization of the attribute columns effectively destroys any pattern that might be in the data, so this gives us a baseline for comparing the current clustering with random data. This will be clarified with a concrete example. At the moment, the stopping point is the level at which the difference between the observed data and the randomized data are a maximum.

Within and Between Cluster Dissimilarities There are many variations on the theme we are about to play. Suppose we have an input DC d and an output ultrametric u. At level h, $u(h)$ determines a partition of the underlying set E by means of clusters $C_1, C_2, \ldots, C_k$. The average within-clusters DC is the mean of the set of all $d(x, y)$ such that $u(x, y) \leq h$. The average between-clusters DC is the mean of all $d(x, y)$ such that $u(x, y) > h$. The underlying goal of a cluster algorithm is to group together points for which $d(x, y)$ is small and separate points for which $d(x, y)$ is large. Unfortunately this is somewhat of a balancing act. Sometimes decreasing the mean within-cluster distance also decreases the mean between-cluster distance. Different authors have different ways of dealing with this issue.

Local Version of Goodman-Kruskal We want every within cluster distance to be smaller than every between cluster distance. Look at all pairs ab, xy such that a, b are in the same cluster and x, y are in different clusters. Call the pair (ab, xy) concordant if $d(a, b) < d(x, y)$ and discordant if $d(a, b) > d(x, y)$. Ignore ties, and define

S_+, S_- as above. The γ statistic is defined as above, but now we have a value for each fusion level.

We will provide a numerical illustration of all this in connection with Example 5.1.

Example 5.1 This example is taken from the Clustan distribution package, and originally appeared in [13]. It classifies mammals milk from 25 different mammals on the basis of five attributes: water, protein, fat, lactose, ash. The attributes are standardized so that they are represented by Z-scores. The dissimilarity is squared Euclidean distance. We will just present the various tree representations. These can be found at the end of the paper. The example is only presented to illustrate the algorithms presented in the section on validation. Of course it also vividly illustrates the fact that different algorithms produce different clusterings.

The stopping rules will be illustrated for Figure 11. Table 5.1 shows the statistics relative to the 2, 3, 4, 5, 6 cluster levels. These atatistics should only be used for guidance. There is nothing terribly persuasive about them. The Clustan statistics (not illustrated here) all indicate stopping at the level with 3 clusters. In the table, there are 5 columns having the following meanings: *num* represents the number of clusters, *fusion* the corresponding fusion level, *within* the mean within cluster DC, *between* the mean between cluster DC , *ratio* the ratio of the mean between cluster levels and the within cluster levels, and finally we have *gamma* which represents the local version of the Goodman-Kruskal γ-statistic. Table 5.1 also seems to indicate a stopping level of 3 clusters.

num	*fusion*	*within*	*between*	*ratio*	*gamma*
6	0.651	0.228	2.42	10.68	0.892
5	0.858	0.251	2.44	9.72	0.874
4	0.921	0.411	2.89	7.03	0.857
3	1.329	0.528	3.10	5.86	0.826
2	2.976	1.419	5.21	3.67	0.859

Table 5.1: Stopping rules

We close by illustrating the Clustan algorithm for the same data as that used for Table 5.1. It appears in Figure 7. The Figure will be explained during the lecture.

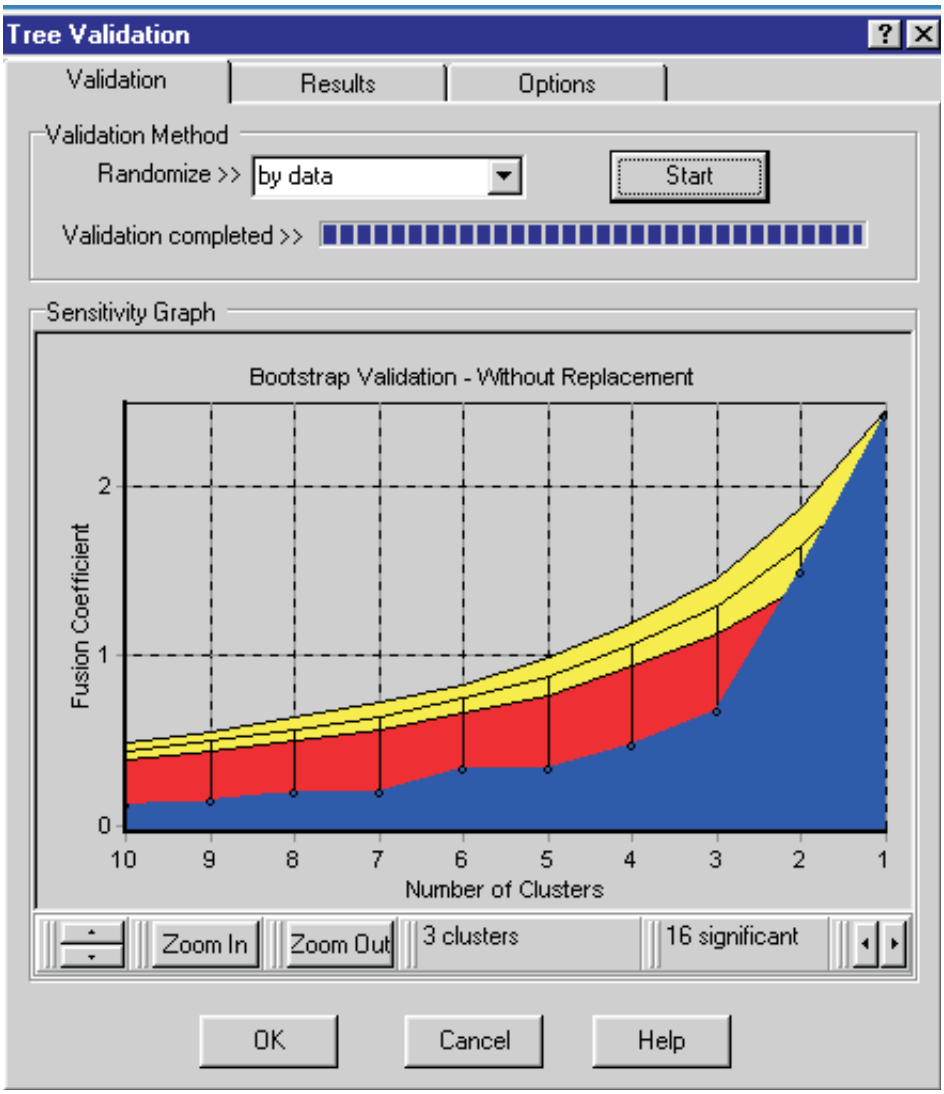

Figure 7: Demonstration of CLUSTAN Validate Algorithm

References

[1] M. R. Anderberg, Cluster Analysis for Applications, Academic Press, 1973.

[2] F. B. Baulieu, A Classification of Presence/Absence based Dissimilarity Coefficients, *J. of Classification*, **6**, pp. 233-246, 1989.

[3] F. B. Baulieu, Two Variant Axiom Systems for Presence/Absence based Dissimilarity Coefficients, *J. of Classification*, **14**, pp. 159-170, 1997.

[4] H. T. Clifford and W. Stephenson, An Introduction to Numerical Classification, Academic Press, New York, 1975.

[5] G. D. Crown and M. F. Janowitz, *The Controversy About Continuity in Clustering Algorithms*, *DIMACS Technical Report **2002-004***, Piscataway, NJ, 2002.

[6] W. H. E. Day and H. Edelsbrunner, Efficient algorithms for agglomerative hierarchical clustering methods, *J. of Classification*, **1**, pp. 7–24, 1984.

[7] W. H. E. Day and H. Edelsbrunner, Investigations of proportional link linkage clustering methods, *J. of Classification*, **2**, pp. 239–254, 1985.

[8] B. S. Duran and P. L. Odell, Cluster Analysis. A Survey., Lecture Notes in Economics and Mathematical Systems **100**, Springer-Verlag, Berlin, 1974.

[9] B. Everitt, S. Landau, and M. Leese, Cluster Analysis (4th ed.), Arnold, London, 2001.

[10] L. Fisher and J. W. Van Ness. Admissible clustering proceedures, *Biometrika* **58**, PP. 91–104, 1971.

[11] A. D. Gordon, Classification (2nd ed.), Chapman & Hall, London, 1999.

[12] J. C. Gower and P. Legendre, Metric and Euclidean Properties of Dissimilarity Coefficients, *J. of Classification*, **3**, pp. 5-48, 1986.

[13] J. A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.

[14] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs 1988.

[15] M. F. Janowitz, Continuous L-Cluster Methods, *Discrete Applied Mathematics* **3** (1981), 107–112.

[16] N. Jardine and R. Sibson, Mathematical Taxonomy, Wiley, New York, 1971.

[17] L. Kaufman and P. J. Rousseeuw, Finding Groups in Data. An Introduction to Cluster Analysis, Wiley, New York, 1990.

[18] G. Lance and W. T. Williams, A general theory of classificatory sorting strategies: II, *Computer Journal* **10**, pp. 271-277, 1967.

[19] I. G. Lerman, Les bases de la Classification Automatique, Gauthier-Villars, Pairs, 1970.

[20] P. Legendre and L. Legendre, Numerical Ecology, Second English ed., Elsevier, Amsterdam, 1998.

[21] J. E. Mezzich and H. Solomon, Taxonomy and Behavioral Science, Academic Press, New York, 1980.

[22] G. W. Milligan, A Monte Carlo study of thirty internal criterion measures for cluster analysis, *Psychometrika* **46**, pp. 187-199, 1981.

[23] B. Mirkin, Mathematical; Classification and Clustering, Kluwer, Dordrecht, 1996.

[24] R. Mojena, Hierarchical grouping methods, *The Computer Journal* **20**, pp. 359–363, 1977.

[25] R. Mojena and D. Wishart, Stopping rules for Ward's clustering method, in *Proceedings of COMPSTAT 1980*, Physica-Verlag, Würzburg, pp. 426–432, 1980.

[26] R. R. Sokal and P. H. A. Sneath, Principles of Numerical Taxonomy, Freeman, San Francisco, 1963.

[27] P. H. A. Sneath and R. R. Sokal, Numerical Taxonomy, Freeman, San Francisco, 1973.

[28] H. Späth. Cluster Analysis Algorithms for data reduction and classification of objects, Ellis Horwood, Chichester, 1980.

[29] J. Van Ryzin (ed.) Classification and Clustering, Academic Press, New York, 1977.

[30] J. H. Ward, Jr., Hierarcical grouping to optimize an objective function, *Journal of the American Statistical Association* **58**, pp. 236–244, 1963.

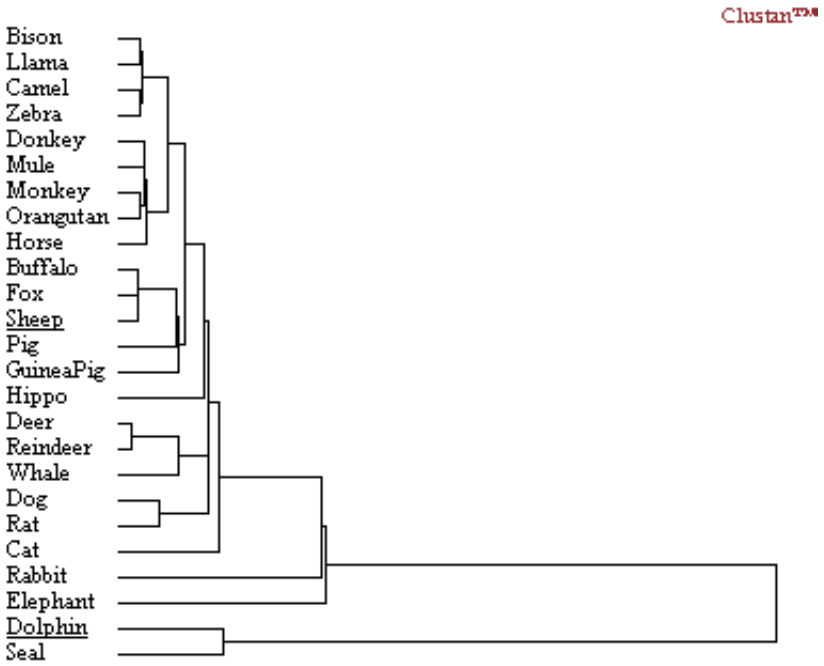

Figure 8: Single Linkage Clustering on Mammal Data

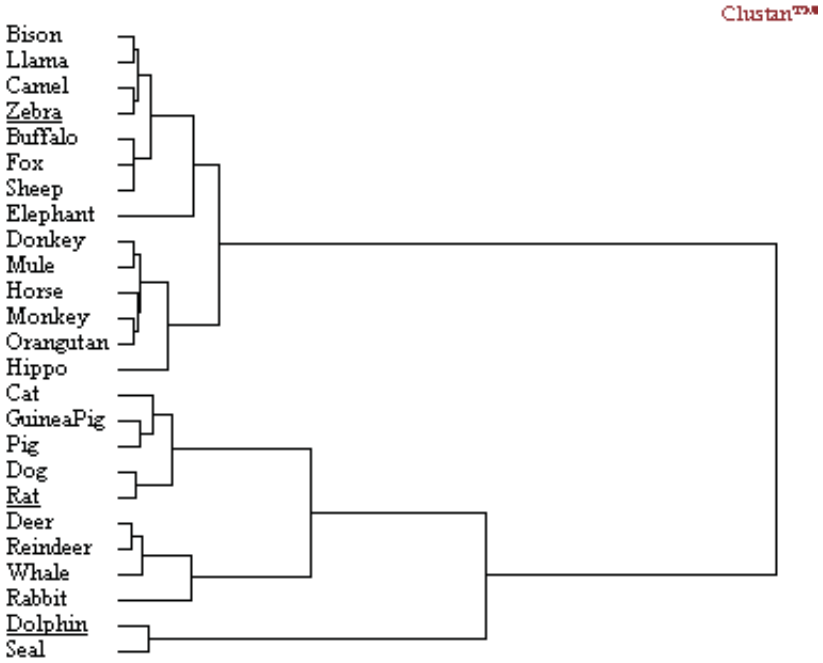

Figure 9: Complete Linkage Clustering on Mammal Data

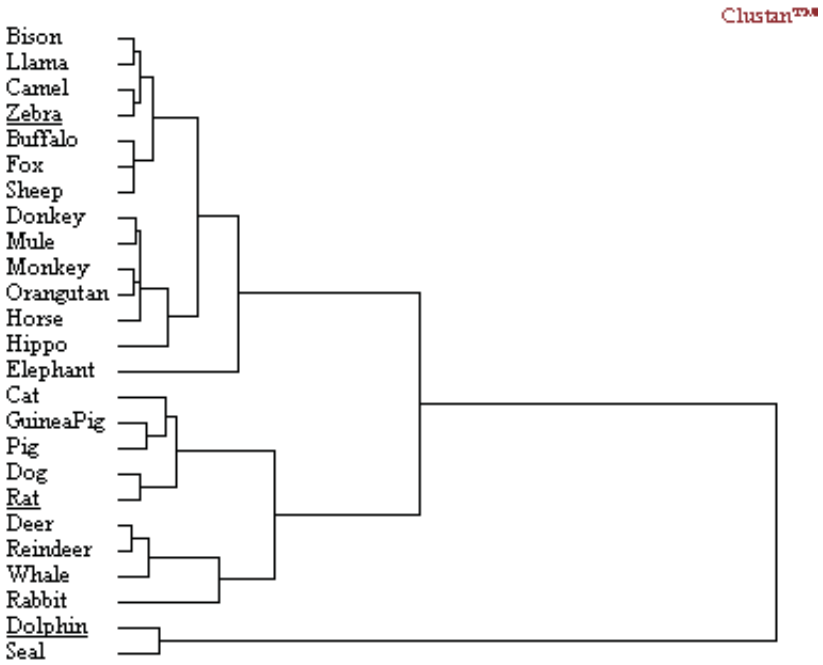

Figure 10: UPGMA Clustering on Mammal Data

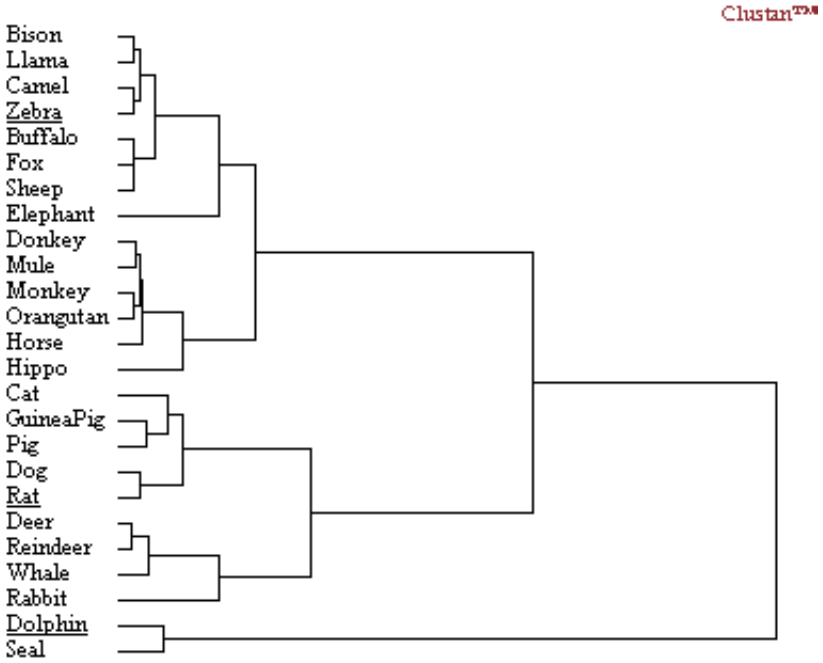

Figure 11: WPGMA Clustering on Mammal Data

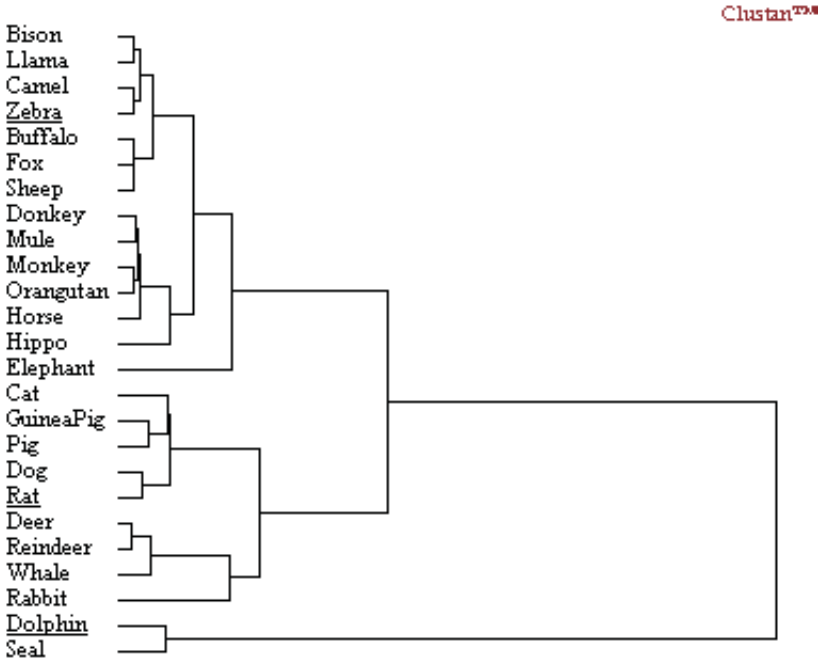

Figure 12: UPGMC Clustering on Mammal Data

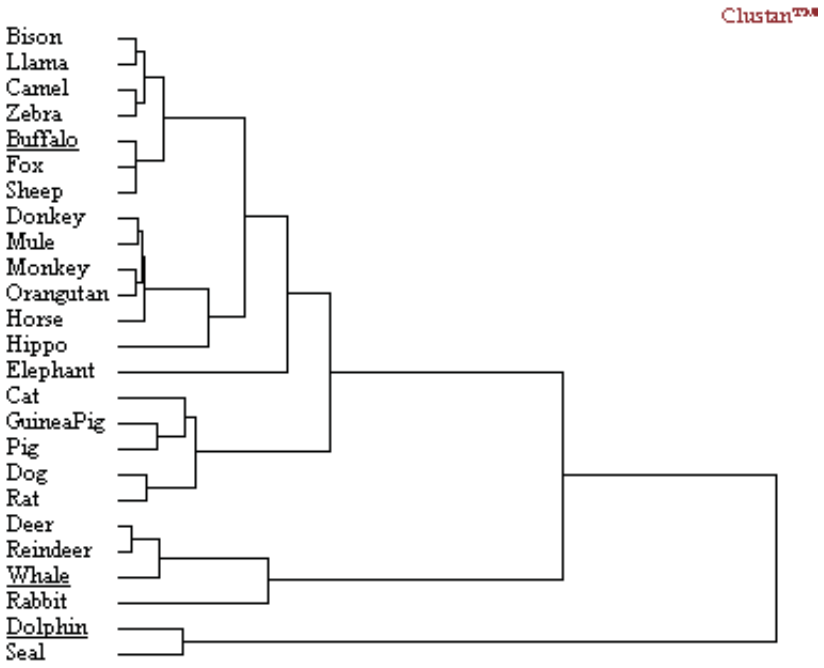

Figure 13: WPGMC on Mammal Data

A Appendix

The CSNA web site has a number of references to commercial software, as does [9], pp. 197-207. In these notes we used Clustan. Their web site is

`http://www.clustan.com`,

while the CSNA site is at

`http://www.pitt.edu/~csna`

A number of text books are mentioned in the references:
[1, 4, 8, 9, 11, 13, 14, 16, 17, 19, 20, 21, 23, 26, 27, 28, 29]