Cluster Analysis

Hermann Moisl

Abstract Cluster analysis comprises a collection of mathematically-based computational methods for identifying structure in data that is too large or too complex, or both, to be easily interpreted by direct inspection. Its main use in corpus linguistics is hypothesis generation: any structural regularities that cluster analysis finds in data abstracted from a corpus are used to draw inferences on the basis of which one or more hypotheses about the corpus content can be formulated. After motivating the use of cluster analysis with an example, the fundamentals of cluster analysis are introduced, including the mathematical concepts of vector space and proximity in vector space. Two of the most frequently used methods, *k*-means and hierarchical analysis, are then described, followed by outlines of two representative case studies. The discussion concludes with a guide to cluster analysis using R, advice on how clustering results should be reported, and some additional readings.

18.1 Introduction

The advent of digital technology has generated a rapid increase in the number and size of corpora available to the linguist, and data abstracted from these can be so complex as to be impenetrable to understanding by direct inspection. The aim of this chapter is to outline how cluster analysis can be used for interpretation of such data. Cluster analysis is primarily a tool for hypothesis generation. It identifies structure that is latent in data, awareness of which can be used to draw inferences on the basis of which a hypothesis is formulated. The discussion first presents the fundamentals of cluster analysis and describes two of the most frequently used methods, then briefly outlines two representative clustering applications, and finally presents a guide to cluster analysis of linguistic data using R together with recommendations for presentation of clustering results and some additional reading.

The use of mathematical and statistical methods has historically been unevenly distributed among research communities and, within them, across linguistics subfields. In the USA and Britain the dominance of generative linguistics, with its emphasis on competence modelling, has until fairly recently discouraged the development of such methods, whereas continental Europe has maintained a strong tradition of quantitative linguistics throughout the twentieth century (Köhler and Hoffmann 1995). Recent decades have, however, seen a vigorous growth in the use of quantitative methods in linguistics research worldwide. Cluster analysis in particular is now used in synchronic grammatical and in historical, geographical, and social variation research as well as in language processing technologies such as information extraction, question answering, machine translation, and text type identification. The by-now extensive associated literatures up to 2015 are summarized, with references, in Moisl (2015, ch.6); sample references for synchronic grammar are Korhonen (2010), Hauer and Kondrak (2011), Gries (2010a,b); for historical linguistics Kessler (2008), Delmestri and Cristianini (2012); for dialectology Wieling et al. (2011), Meschenmoser and Pröll (2012); and for sociolinguistics Ruette et al. (2013), Grieve et al. (2011).

Hermann Moisl
Newcastle University, UK
hermann.moisl@ncl.ac.uk

18.2 Fundamentals

18.2.1 Motivation

Assume a sociolinguist wants to understand the nature of phonetic usage in some speech community of interest, that is, whether there is systematic rather than random variation among speakers. A representative corpus of speech is collected, a set of phonetic variables is defined, and the number of times each speaker uses each of these variables is recorded, thereby building up a body of data. Table 18.1 shows the number of times each of 24 randomly selected speakers, one per row, uses each of the 12 phonetic variables in the columns.

Table 18.1 Frequency of use of 12 phonetic variables by selected speakers

Speaker	ə$_1$	ə$_2$	**o:**	ə$_3$	**ï**	**eï**	**n**	**a:$_1$**	**a:$_2$**	**aï**	**r**	**w**
g01	3	1	55	101	33	26	193	64	1	8	54	96
g02	8	0	11	82	31	44	205	54	64	8	83	88
g03	4	1	52	109	38	25	193	60	15	3	59	101
n01	100	116	5	17	75	0	179	63	0	19	46	62
g04	15	0	12	75	21	23	186	57	6	12	32	97
g05	14	6	45	70	49	0	188	40	0	45	72	79
g06	5	0	40	70	32	22	183	46	0	2	37	117
n02	103	93	7	5	87	27	241	52	0	1	19	72
g07	5	0	11	58	44	31	195	87	12	4	28	93
g08	3	0	44	63	31	44	140	47	0	5	43	106
g09	5	0	30	103	68	10	177	35	0	33	52	96
g10	6	0	89	61	20	33	177	37	0	4	63	97
n03	142	107	2	15	94	0	234	15	0	25	28	118
n04	110	120	0	21	100	0	237	4	0	61	21	62
g11	3	0	61	55	27	19	205	88	0	4	47	94
g12	2	0	9	42	43	41	213	39	31	5	58	124
g52	11	1	29	75	34	22	206	46	0	29	34	93
g53	6	0	49	66	41	32	177	52	9	1	68	74
n05	145	102	4	6	100	0	208	51	0	22	61	104
n06	109	107	0	7	111	0	220	38	0	26	19	70
g54	3	0	8	81	22	27	239	30	32	8	80	116
g55	7	0	12	57	37	20	187	77	41	4	58	101
g56	12	0	21	59	31	40	164	52	17	6	45	103
n07	104	93	0	11	108	0	194	5	0	66	33	69

What hypothesis would one formulate from visual inspection of this data, taking into account all the variables? And what about, say, 100 speakers and 150 variables? These questions are clearly rhetorical, and there is a moral: human cognition is unsuited to seeing regularities in anything but the smallest collections of numerical data. To see the regularities we need help, and that is what cluster analysis provides: it is a family of computational methods for identification and graphical display of structure in data when the data are too large either in terms of the number of variables or of the number of objects described, or both, to be readily interpretable by direct inspection.

There are numerous clustering methods. A frequently used one is hierarchical analysis, described later in this discussion. Hierarchical analysis of the data in Table 18.1 yields the following result.

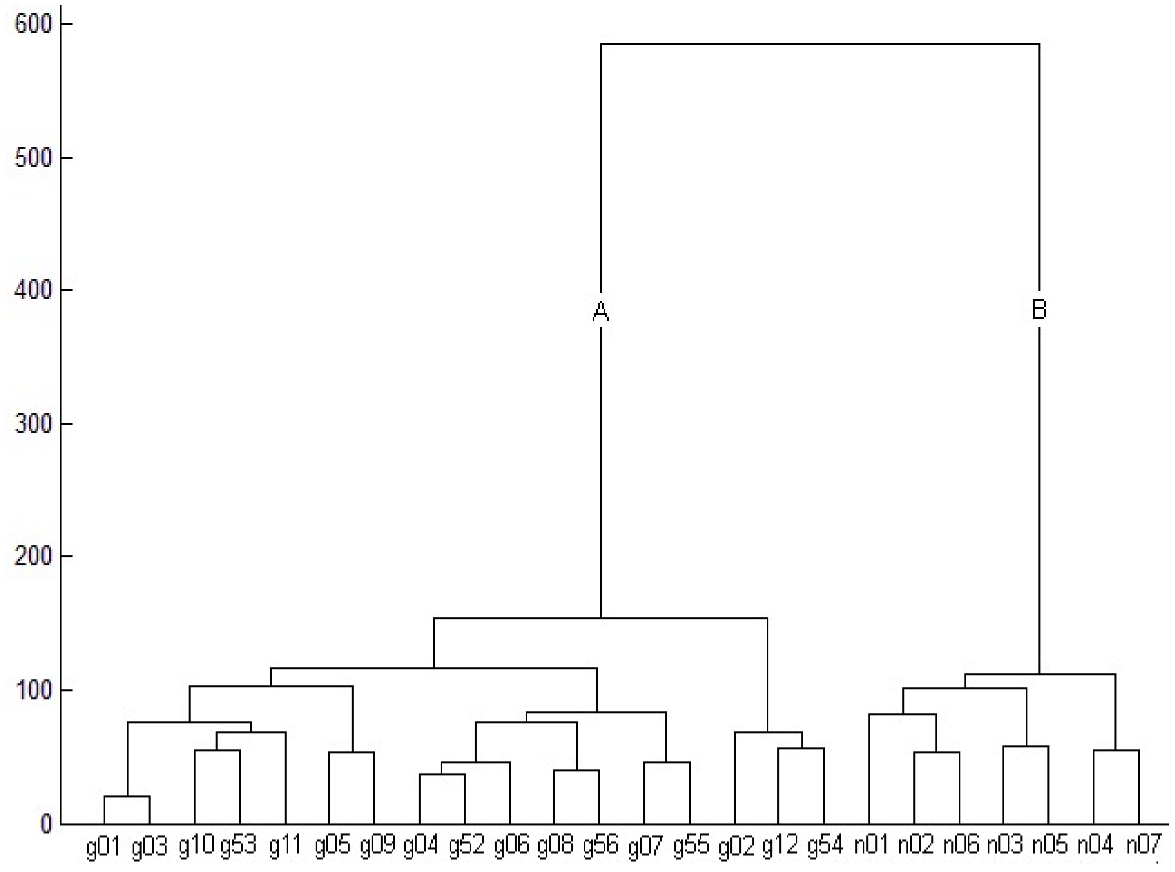


Fig. 18.1 Hierarchical clustering of row vectors in Table 18.1

Hierarchical cluster analysis is so called because it arranges data objects into a hierarchy of relative similarities. Each row in Table 18.1 is a profile of a speaker's phonetic usage as described by the frequency of the various phonetic variables, and visual inspection of the table will show that some profiles are more similar to one another than to others. Hierarchical analysis uses these relative degrees of similarity to group the profiles into clusters and to display the cluster structure as a tree. How a tree such as the one in Fig.18.1 is constructed will be described later on. For the moment it suffices to observe that similarity between cluster subtrees is represented by the lengths of the vertical arcs joining them: the shorter the arcs the more similar the clusters. In Fig. 18.1 the arcs labelled A and B are much longer than any others in the tree, which shows that the speaker profiles fall into two clusters, where all the profiles in cluster A are more similar to one another than to any in cluster B and vice versa. A and B can be further decomposed into subclusters, so that, for example cluster A consists of subclusters C and D, C consists of E and F, and so on. The complete tree gives an exhaustive picture of the similarity structure of the phonetic profiles in Table 18.1. Based on Fig.18.1, a reasonable hypothesis is that there is systematic phonetic variation in the speech community, and more specifically that the speakers who constitute that community fall into two main groups.

18.2.2 Data

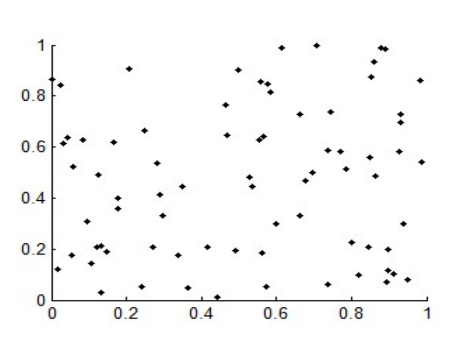
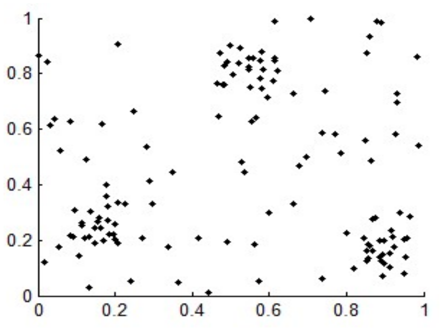
The validity of cluster analysis depends crucially on the characteristics of the data to which it is applied. Data are constructed from observation of things in the world, and the process of construction raises a range of issues that determine the amenability of the data to analysis and the interpretability of the analytical results. On the one hand, nothing can be discovered that is beyond the limits of what the data say about the world. On the other, failure to understand and where necessary to emend relevant characteristics of data can lead to results and interpretations that are distorted or even worthless. It is, therefore, crucial to ensure that, prior to applying any form of cluster analysis, the various data preparation issues have been addressed. These issues cannot be dealt with here, but are covered in detail in Moisl (2015:Chap. 3).

18.2.3 Clustering

This section first discusses the nature of clusters, then presents a geometric concept fundamental to most types of clustering, and finally describes two of the most frequently used methods, *k*-means and hierarchical analysis.

18.2.3.1 Cluster definition

Human perception is optimised to detect patterning in the environment, and clusters are a kind of pattern. Contemplation of a rural scene reveals clusters of trees, of farm buildings, of sheep. Looking up at the night sky reveals clusters of stars, and a casual observer looking at the scatterplots in Fig. 18.2 would say that 2a shows a few small concentrations of points but is essentially random, that 2b has three identifiable clusters of roughly equal size, that 2c has two clusters of unequal size, and that 2d has two intertwined, semi-circular clusters, all embedded in a random scatter of points. That observer would, moreover, have been able to make these identifications solely on the basis of innate pattern recognition capability and without recourse to any explicit definition of the concept 'cluster'.

	
a	b

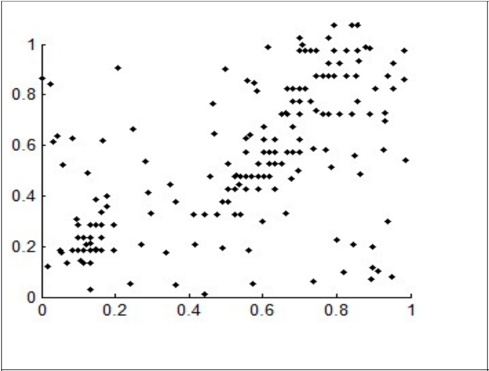
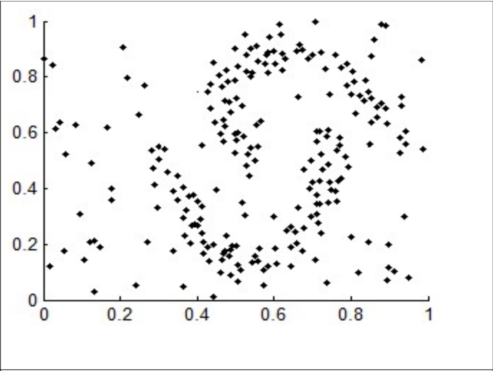
	
c	d

Fig. 18.2 A selection of clusters in a two-dimensional space

Direct perception of patterns is the intuitive basis for understanding what a cluster is, and is fundamental in identifying the cluster structure of data, but it has two main limitations. One limitation is subjectivity and consequent unreliability. This subjectivity stems from the cognitive context in which a data distribution is interpreted: the casual observer brings nothing to the observation but innate capability, whereas the researcher who compiled the data and knows what the distribution represents brings prior knowledge which potentially and perhaps inevitably affects interpretation. In Fig. 18.2c, for example, does the larger cluster on the upper right contain two subclusters? What would the answer be if it were known that the points represent cats in the upper part of the cluster and dogs in the lower? The other limitation is that reliance on innate perceptual capability for cluster identification is confined to what can be perceived, and in the case of data this means a maximum dimensionality of 3 or less for graphical representation; there is no way of directly perceiving clusters in data with dimensionality higher than that.

The obvious way to address these limitations is by unambiguous definition of what a cluster is, relative to which criteria for cluster membership can be stated and used to test perceptually-based intuition on the one hand and to identify non-visualisable clusters in higher-dimensional data on the other. Textbook discussions of cluster analysis uniformly agree, however, that no one has thus far succeeded in formulating such a definition. In principle, this lack deprives cluster analysis of a secure theoretical foundation. In practice, the consensus is that there are intuitions which, when implemented in clustering methods, give conceptually useful results, and it is on these intuitions that contemporary cluster analysis is built.

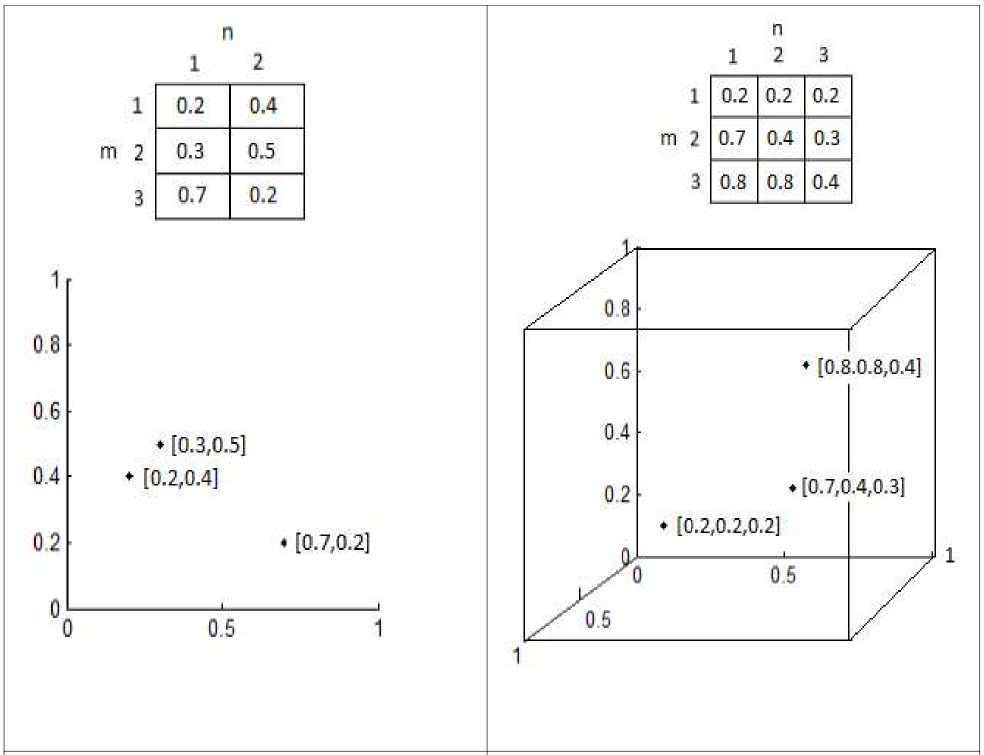
The fundamental intuition underlying cluster analysis is that data distributions contain clusters when the data objects can be partitioned into groups on the basis of their relative similarity such that the objects in any group are more similar to one another than they are to objects in other groups, given some definition of similarity. The most commonly used similarity definition is based on the concept of proximity in vector space.

18.2.3.2 Proximity in Vector Space

Once data are represented as a matrix like that in Table 18.1, they have a geometrical interpretation in terms of the concept of vector space. Given a matrix M with *m* rows and *n* columns:

- The dimensionality of M, that is, the number *n* of columns representing the *n* data variables, defines an *n*-dimensional data space.
- Each of the *m* row vectors is a point in that space.
- The sequence of *n* numbers comprising each row vector specifies the coordinates of the point in the space.

This is shown for a two and three dimensional vector spaces in Fig. 18.3.



a. A two-dimensional vector space containing three vectors

b. A three-dimensional vector space containing three vectors

Fig. 18.3 Vector spaces with corresponding matrices

This idea extends to any dimensionality *n*. For *n* = 4 the analogy between mathematical and physical space breaks down in that four and higher dimensional spaces can neither be conceptualised nor represented in terms of physical space. Mathematically and geometrically, however, higher-dimensional spaces are defined and used in the same way as the foregoing lower-dimensional ones, so that a four-dimensional vector is a point in four-dimensional space, a five-dimensional vector is a point in five-dimensional space, and so on for any *n*. The important thing to realise is that the intuitive concepts of physical space and dimension are useful metaphors for interpreting the corresponding mathematical concepts, but that the metaphors do not constrain what is possible mathematically.

Given a distribution of vectors in a space V, the geometrical proximity of two vectors **v1** and **v2** in the space can be measured by determining the angle between them (Moisl 2015:28-47), but is most often done by measuring the distance separating them in the space. There are numerous measures (Deza and Deza 2009), but by far the most often used is Euclidean one, which is the shortest distance between points. It is calculated using the Pythagorean formula for finding the length of the hypotenuse of a right-angled triangle, as in Fig. 18.4.

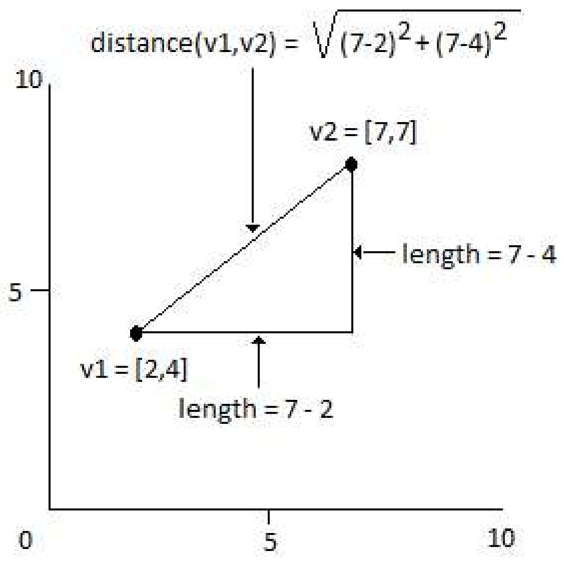


Fig. 18.4 Finding the Euclidean distance between two vectors **v1** and **v2**

18.2.3.3 Clustering Methods

Clustering methods are standardly divided into two types: (i) non-hierarchical, which partition data vectors into some number *k* of disjoint groups such that the members of any given group are closer to one another in the data space than they are to members of any other group, and (ii) hierarchical, which show the distance relations among data vectors as a tree and leave the cluster structure to be inferred from the tree by the researcher. Because these types offer complementary information about the structure of data, examples of both are described here, starting with a non-hierarchical one.

K-means

K-means clustering is the most frequently used non-hierarchical clustering method. The following account first describes the standard *k*-means algorithm and then identifies issues associated with it.

K-means is based on the idea that, for a given set of vectors V, each cluster is represented by a prototype vector, and a cluster is defined as the subset of vectors in V which are in distance terms closer to the prototype than they are to the prototype of any other cluster. A mathematical function known as an objective function is used to find a set of clusters each of which optimally meets this criterion; more is said about the form of such a function below. For V comprising *m* *n*-dimensional vectors, V is partitioned into *k* prototype-centred clusters by the following iterative procedure:

1. Initialise by selecting *k* *n*-dimensional prototype locations in the vector space; these can be anywhere in the space. The prototypes are the initial estimates of where the clusters are centred in the space, and their locations are refined in subsequent steps. Placement of initial prototypes and selection of a value for *k*, that is, of the number of clusters, is non-trivial, and is further discussed below.

2. Assign each of the *m* data vectors to whichever of the *k* prototypes it is closest to in the space using a suitable distance measure such as the Euclidean. This yields *k* clusters.

3. Calculate the centroid of each of the *k* clusters resulting from (2), where 'centroid' is here understood as the centre of a distribution of vectors in *n*-dimensional space. The centroid of a set of row vectors in a matrix M is calculated by taking the mean of the column vectors of M, as exemplified in Table 18.2. Each centroid becomes a new cluster prototype.

Table 18.2 Calculating the centroid of a set of vectors

		variable 1	**variable 2**	**variable 3**
M	**V_1**	0.1	0.2	0.3
	V_2	0.4	0.5	0.6
	V_3	0.7	0.8	0.9
	V_4	0.9	0.8	0.7
	V_5	0.6	0.5	0.4
	V_6	0.3	0.2	0.1
Centroid		0.5	0.5	0.5

4. Repeat (2) and (3) until the objective function is optimised, that is, until the centroids stop changing their locations in the space.

This procedure is visualised in Fig. 18.5 for 29 data points in a two dimensional space, though it extends to any dimensionality. There are three visually obvious clusters, labelled A – C, and the aim is for the above procedure to find them by identifying the cluster centroids in a way that optimises the objective function.

INSERT FIGURE 5a	INSERT FIGURE 5b
a	b
INSERT FIGURE 5c	INSERT FIGURE 5d
c	d
INSERT FIGURE 5e	INSERT FIGURE 5f
e	f

Fig. 18.5 Identification of clusters using the *k*-means procedure

Figure 18.5a shows a scatter plot of the 29 data points and $k = 3$ prototypes randomly selected in the space and represented as crosses. The points are assigned to their nearest prototypes, which results in the clusters enclosed by the rectangles in Fig. 18.5b. Cluster centroids are now calculated; their positions, shown in 5c, reflect the means of the data points in the clusters of 5b, and they become the new prototypes. Because the positions of the prototypes have changed, so does the clustering, as shown in 5d. The cluster centroids are once again calculated; their new positions are shown in 5e and the associated clustering in 5f. Further iterations will not change the positions of the centroids, and the procedure terminates. The data points associated with the final prototypes correspond to the visually identifiable clusters in Fig. 18.5a, and the *k*-means procedure can therefore be said to have partitioned the data set into three disjoint subsets. The further claim is that, for $k = 3$, the partition is optimal in the sense that it has minimised an objective function. Where Euclidean distance is used, the objective function is usually the sum of squared errors (SSE):

$$SSE = \Sigma_{i = 1..k} \Sigma_{x \in C_i} (x - p_i)^2 \qquad (18.1)$$

where x is a data point, C_i is the i'th of k clusters, and p_i is the prototype of the i'th cluster. This expression says that the SSE is the sum, for all k clusters, of the Euclidean distances between the cluster prototypes and the data points associated with each prototype. For *k*-means to have optimised this function, the prototypes have to be placed in the data space so that the Euclidean distances between them and their associated data points is globally minimised across all clusters. It is easy to see that this is what *k*-means does: the procedure converges on stable cluster centroids, and a centroid is by definition the minimum distance from all the points on which it is based.

The *k*-means algorithm is easy to understand and its results easy to interpret, it is theoretically well founded in linear algebra, and its effectiveness has repeatedly been empirically demonstrated in research applications. It also has well known problems, however; the main ones are outlined below.

- Selection of initial cluster centroids
  K-means requires the user to specify the locations of the initial k prototypes $c_1...c_k$ in the data space, but different sets of initial prototypes can lead to different final ones and thereby to different partitions of the data into clusters. In Fig. 18.6 initial prototypes are shown as filled circles and final ones as crosses; given the correct $k = 3$, one initialisation led to a clustering compatible with visual intuition, and the other did not.

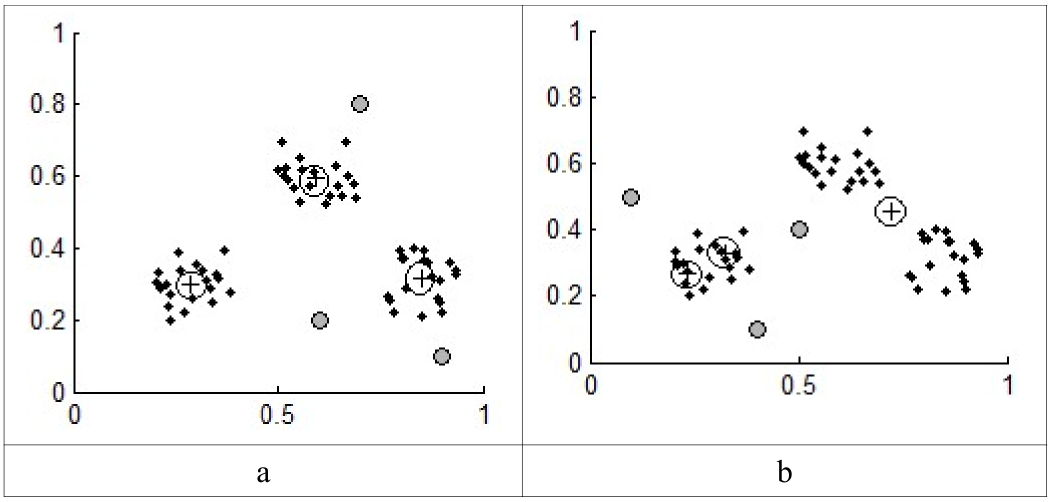


Fig. 18.6 Effect of initial prototype location on *k*-means cluster solutions

The standard solution is to conduct a succession of analyses on the same data using a different, usually randomly-generated set of initial prototypes for each analysis, keeping a record of the SSE value in each case. The best analysis, that is, the one with the optimal SSE, is then selected.

- How many clusters?
  K-means requires the user to specify the number of clusters *k* to identify in the data. If the value chosen for *k* is incompatible with the number of clusters the data actually contains, however, the result will be misleading because *k*-means will deliver *k* clusters whatever the actual number of clusters intrinsic to the data, including none. For example, Fig. 18.7 shows the cluster structure from Fig. 18.6, but with $k = 2$ and $k = 4$: in both cases *k*-means fails to identify the visually-clear cluster structure.

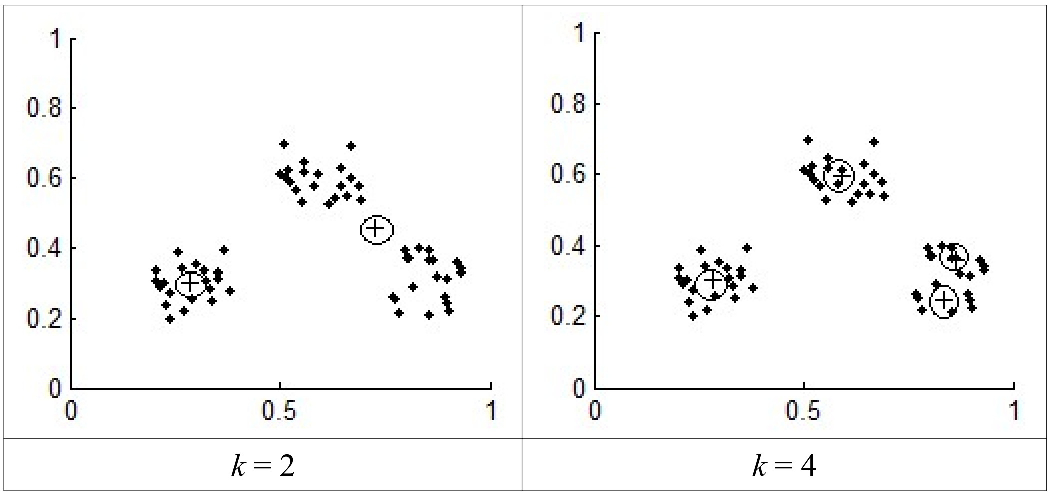


Fig. 18.7 *K*-means cluster solutions for $k = 2$ and $k = 4$

  The obvious solution is to base the selection of *k* on reliable *a priori* knowledge of the domain from which the data comes, where available, but this obviates the point of the exercise, which is to identify unknown structure in data. Failing this, some other clustering method such as the one covered next can be used to gain insight into its cluster structure, or one of the range of initialisation heuristics proposed in the literature can be applied (Xu and Wunsch 2009:Chap. 4.3; Mirkin 2011:Chap. 6.2.7). A frequently used heuristic is to conduct multiple analyses on the same data using different values of *k* and to select the best one, given some definition of "best" such as SSE: the value of *k* which optimises SSE is the one selected.

  Selection of *k* and selection of prototypes for any given *k* are obviously interrelated and should therefore be done in conjunction (see Sect. 18.3 for how this can be done).

- Cluster shape
  K-means is limited in the shapes of clusters it can identify. This is demonstrated with reference to the clusters in Fig. 18.8.

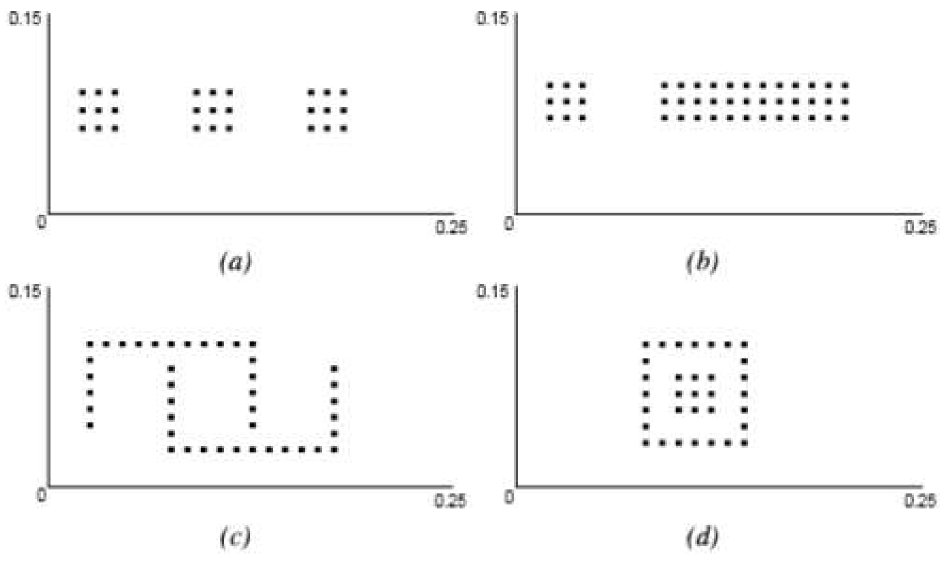


Fig. 18.8 A range of cluster shapes for *k*-means analysis

The *k*-means analysis of the data underlying Fig. 18.8a-d gave the results shown in Fig. 18.9.

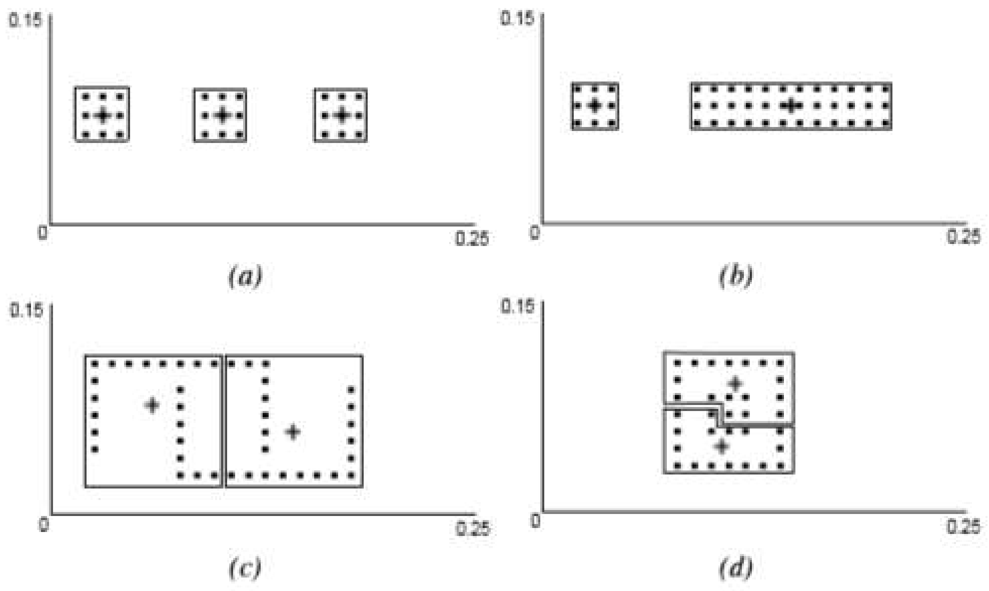


Fig. 18.9 *K*-means solutions for the clusters in Fig. 18.8

In Fig. 18.9a–d the consensus prototypes arrived at by multiple initialisations of *k*-means are shown by crosses, and the data points associated with each prototype are enclosed by boxes. The partitions of 9a and 9b accord with visual intuitions about cluster structure. However, those of 9c and 9d do not; this failure is symptomatic of a fundamental limitation on the clustering capability of *k*-means, which has to do with linear separability (Moisl 2015:189-201).

- Outliers
  Because *k*-means is based on centroids, it is strongly affected by the presence of

outliers which distort the location of the centroids in the data space. Outliers in data should therefore be identified and eliminated prior to analysis.

Hierarchical clustering

Given an $m \times n$ matrix M which represents m objects in n-dimensional space, hierarchical cluster analysis constructs a tree which represents the distance relations among the m objects in the space, as shown in Fig. 18.1.

Construction of a cluster tree is a two-step process: the first step abstracts a distance table from M, and the second constructs the tree by successive transformations of the table. An intuition for how tree construction proceeds is best gained by working through an example; the example that follows is based on the first 6 rows of the data matrix in Table 18.1. The Euclidean distances between all possible pairings of these 6 rows were calculated and stored in a 6×6 matrix D, shown in Table 18.3.

Table 18.3 Euclidean distance matrix for the first six rows of Table 18.1

	g01	**g02**	**g03**	**g04**	**g05**	**g06**
g01	0	116.9	58.9	82.6	103.8	69.7
g02	116.9	0	113.2	98.8	124.4	116.8
g03	58.9	113.2	0	79.4	112.9	69.3
g04	82.6	98.8	79.4	0	108.8	78.9
g05	103.8	124.4	112.9	108.8	0	112.9
g06	69.7	116.8	69.3	78.9	112.9	0

The Euclidean distance from g01 to itself is 0, g01 to g02 is 116.9, g01 to g03 is 58.9, and so on. To simplify the discussion to follow, it is observed that the table is symmetrical on either side of the zero-values on the main diagonal because the distance between any two row vectors in the data matrix is symmetrical --the distance from g02 to g03 is the same as the distance from g03 to g02. Since the upper-right triangle simply duplicates the lower-left one it can be deleted without loss of information. The result is shown in Table 18.4.

Table 18.4 The Euclidean distance matrix of Table 18.3 with upper triangle values removed

	g01	**g02**	**g03**	**g04**	**g05**	**g06**
g01	**0**					
g02	116.9	0				
g03	58.9	113.2	0			
g04	82.6	98.8	79.4	0		
g05	103.8	124.4	112.9	108.8	0	
g06	69.7	116.8	69.3	78.9	112.9	0

Given *m* objects to be clustered, construction of a tree begins with *m* clusters each of which contains a different single object. Thereafter, the tree is constructed in a sequence of steps in which, at each step, two clusters are joined into a superordinate cluster and the distance matrix D is transformed so as to incorporate the newly created cluster into it. The sequence ends when only one cluster, the tree itself, remains and D is empty. The joining of clusters requires some criterion for deciding which of the clusters available at any given step in the tree construction process should be selected for combination. The literature contains a variety of different criteria, and some of these will be presented below; the one chosen for detailed description joins the two clusters with the smallest distance between them in the distance matrix.

The following sequence of cluster joins and matrix transformations exemplifies this. Initially, each row vector of the data matrix is taken to be a cluster on its own. Fig. 18.10a shows the original distance matrix of Table 18.4. It is searched to find the smallest distance between clusters. This is the distance 58.9 between column 1, that is, g01 and row 3, that is, g03, shown bold-face. These are combined into a new composite cluster (1,3).

	g01	g02	g03	g04	g05	g06
g01	0					
g02	116.9	0				
g03	**58.9**	113.2	0			
g04	82.6	98.8	79.4	0		
g05	103.8	124.4	112.9	108.8	0	
g06	69.7	116.8	69.3	78.9	112.9	0

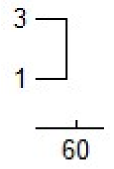
a: Initial distance table reproduced from Table 18.4, with smallest distance highlighted

	(1,3)	2	4	5	6	
(1,3)	0					
2	**113.2**	0				
4	**79.4**	98.8	0			
5	**103.8**	124.4	108.8	0		
6	**69.3**	116.8	78.9	112.9	0	

b: Transformed version of (a)

Cluster 1	Cluster 2	Joining distance
1	3	58.9

c: Table of joins



d: Graphical representation of (c)

Fig. 18.10 Joining the two nearest single-speaker clusters into a composite cluster

Figure 18.10a is now transformed into the one in 10b. Rows and columns (1) and (3) are removed from 10a and replaced in 10b with a single blank row and column to accommodate the new (1,3) cluster; 0 is inserted as the distance from (1,3) to itself. Into the blank cells of the (1,3) row and column of Fig. 18.10b are inserted the minimum distances from (1,3) to the remaining clusters (2), (4), (5), and (6). What does this mean? Referring to Fig. 18.10a, the distance between (1) and (2) is 116.9 and between (3) and (2) it is 113.2; the minimum is 113.2, and that value is inserted into the cell representing the minimum distance between (1,3) and (2) in Fig. 18.10b. The distance between (1) and (4) in 10a is 82.6 and between (3) and (4) is 79.4; the latter value is inserted into 10b as the minimum distance between (1,3) and (4). By the same procedure, the minimum distances between (1,3) and 5 and between (1,3) and 6 are inserted. The resulting table is smaller by one row and one column; inserted values are shown in bold-face, and the remaining ones are unchanged. Figure 18.10c is a list showing the sequence of cluster joins together with the distance at which they were combined; this list is the basis for the graphical representation of the cluster tree shown in 10d. The scale line below the tree in 10d allows the approximate joining distance to be read from the graphical representation. One can, for example, see that (1) and (3) are joined just short of 60. The reduced matrix of Fig. 18.10b is used as the basis for the next step, shown in Fig. 18.11a.

	(1,3)	2	4	5	6
(1,3)	0				
2	113.2	0			
4	79.4	98.8	0		
5	103.8	124.4	108.8	0	
6	**69.3**	116.8	78.9	112.9	0

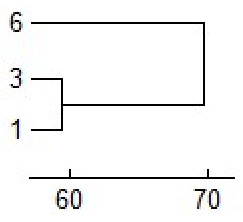
a: Distance table from Fig. 18.10

	((1,3),6)	2	4	5	
((1,3),6)	0				
2	**113.2**	0			
4	**78.9**	98.8	0		
5	**103.8**	124.4	108.8	0	

b: Transformed version of (a)

Cluster 1	Cluster 2	Joining distance
1	3	58.9
(1,3)	6	69.3

c: Table of joins



d: Graphical representation of (c)

Fig. 18.11 Joining the composite cluster in Fig. 18.10b to the nearest single-speaker cluster

The matrix in Fig.18.11a is searched to find the smallest distance between clusters. This is 69.3 between (1,3) and (6), and these are combined into a composite cluster ((1,3),6). The matrix is transformed into the one in 11b as in the previous step: rows and columns (1,3) and (6) are removed and replaced with a single blank row and column to accommodate the new ((1,3),6) cluster, with 0 inserted as the distance from ((1,3),6) to itself. Into the blank cells of the ((1,3),6) row and column are inserted the minimum distance from ((1,3),6) to the remaining clusters (2), (4), and (5). Referring to Fig. 18.11a, the distance between (1,3) and (2) is 113.2 and between (6) and (2) it is 116.8; the minimum is 113.2, and that value is inserted into the cell representing the minimum distance between ((1,3),6) and (2). The distance between (1,3) and (4) is 79.4 and between (6) and (4) is 78.9; the latter value is inserted into the matrix as the minimum distance between ((1,3),6) and (4). By the same procedure, the minimum distance between ((1,3),6) and 5 is inserted. The resulting table is again smaller by one row and one column; inserted values are highlighted, and the remaining ones are again unchanged. Figure 18.11d shows the tree after the second step together with the distance between (1,3) and (6). The reduced matrix of Fig. 18.11b is used as the basis for the next step. The full derivation is given in Moisl (2015:202-08).

For a matrix with *m* rows, there will at any step in the above tree-building sequence be a set of *p* clusters, for *p* in the range 2...*m*, available for joining, and two of these must be selected. At the first step in the clustering sequence, where all the clusters contain a single object, this is unproblematical: simply choose the two clusters with the smallest distance between them.

At subsequent steps in the sequence, however, some criterion for judging relative proximity between composite and singleton cluster pairs or between composite pairs is required, and it is not obvious what the criterion should be. The one exemplified in the foregoing sequence is such a criterion, known as Single Linkage, but there are various others. For simplicity of exposition, it is assumed that a stage in the tree building sequence has been reached where there are $p = 3$ clusters remaining to be joined. This is shown in Fig. 18.12: 12a shows a scatterplot of the data being clustered, and 12b the current state of tree construction.

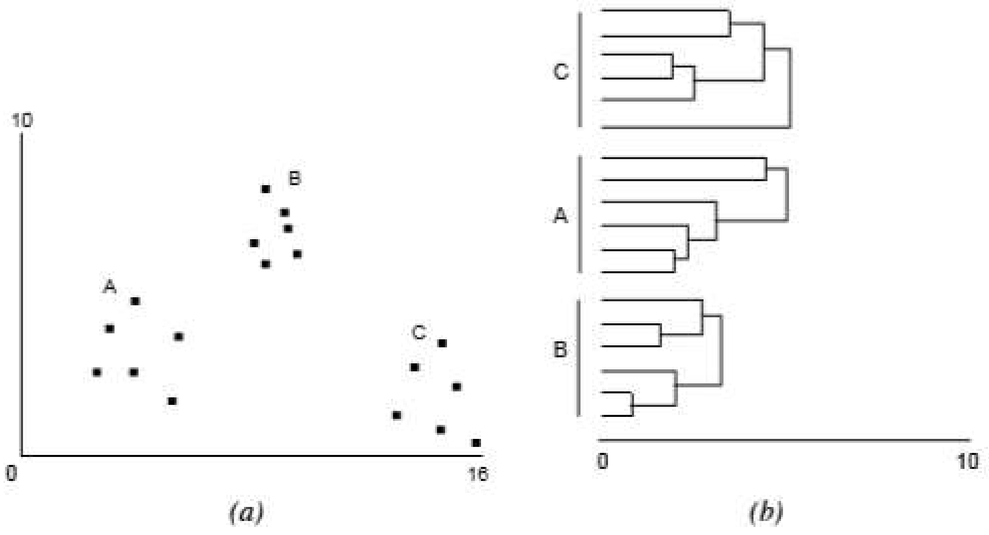


Fig. 18.12 An intermediate stage in a hierarchical cluster tree construction

Which pair of subtrees should be joined next? Based on visual examination of the scatterplot, the intuitively obvious answer is the pair of clusters closest to one another, that is, A and B. Where the data are higher-dimensional and cannot be directly plotted, however, some explicit specification of closeness is required. This is what the following cluster-joining criteria provide.

- The Single Linkage criterion defines the degree of closeness between any pair of clusters (X,Y) as the smallest distance between any of the data points in X and any of the data points in Y: if there are x vectors in X and y vectors in Y, then, for $i = 1...x$, $j = 1...y$, the Single Linkage distance between X and Y is defined as

  $$Single\ Linkage\ Distance(X,Y) = min(dist(X_i, Y_j))$$

  where dist(X_i,Y_j) is the distance between the i'th vector in X and the j'th vector in Y stated in terms of whatever metric is being used, such as Euclidean distance. The Single Linkage distances between all unique pairs of the p vectors remaining to be clustered are calculated, and the pair with the smallest distance is joined. This is exemplified for the three clusters of Fig. 18.12 in Fig. 18.13.

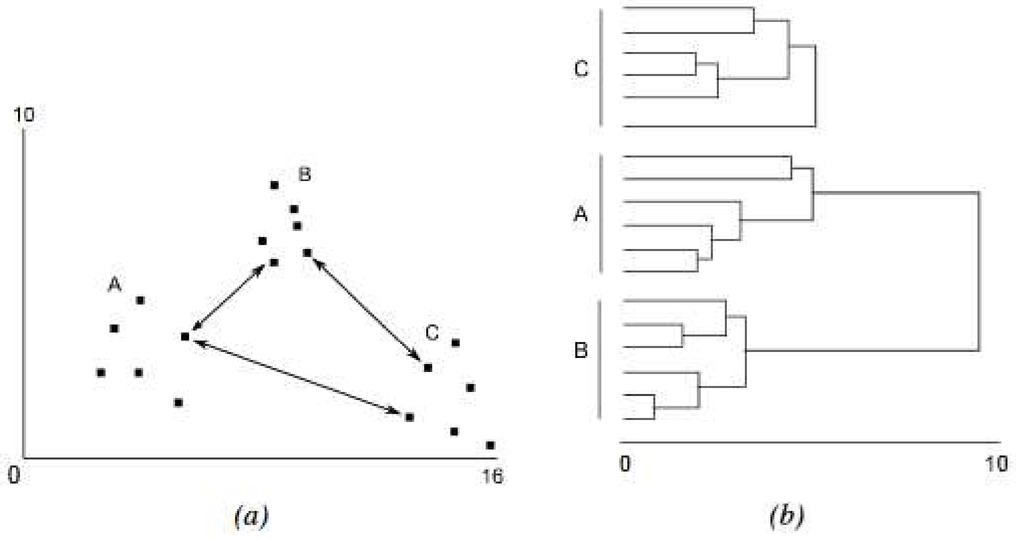


Fig. 18.13 Single Linkage

The arrowed lines in Fig. 18.13a represent distances between the points closest to one another in cluster pairs (A,B), (A,C), and (B,C); the one between A and B is shortest, so these two clusters are joined, as in Fig. 18.13b. Single Linkage is also known as Nearest Neighbour clustering.

- Complete Linkage defines the degree of closeness between any pair of clusters (X,Y) as the largest distance between any of the data points in X and any of the data points in Y: if there are *x* vectors in X and *y* vectors in Y, then, for $i = 1...x$, $j = 1...y$, the Complete Linkage distance between X and Y is defined as

$$\textit{Complete Linkage Distance}(X,Y) = \max(dist(X_i, Y_j))$$

where dist (X_i,Y_j) is the distance between the *i*'th vector in X and the *j*'th vector in Y stated in terms of whatever metric is being used. The Complete Linkage distances between all unique pairs of the *p* vectors remaining to be clustered are calculated, and the pair for which the Complete Linkage distance is smallest is joined. This is exemplified for the three clusters of Fig. 18.12 in Fig. 18.14.

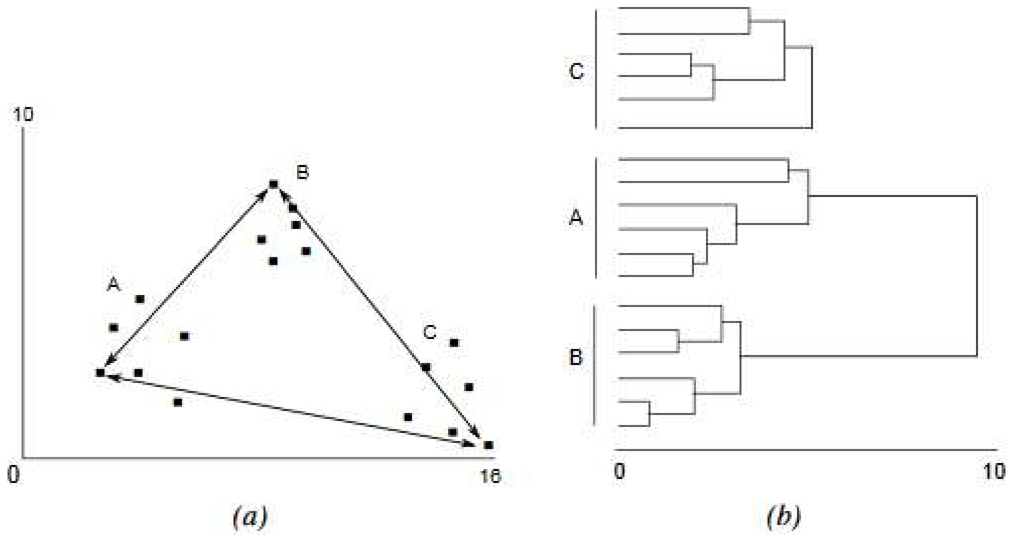


Fig. 18.14 Complete Linkage

The arrowed lines in Fig. 18.14a represent distances between the points furthest from one another in cluster pairs (A,B), (A,C), and (B,C); the one between A and B is shortest, so these two clusters are joined, as in 14b. The intuition behind this joining criterion may not be immediately obvious, but is does make sense: finding and joining the cluster pair with the smallest maximum distance between their members creates a cluster with the smallest diameter at that stage in the clustering procedure, and therefore the most compact cluster. Complete Linkage is also known as Furthest Neighbour clustering.

- The Average Linkage criterion defines the degree of closeness between any pair of clusters (X,Y) as the mean of the distances between all ordered pairs of objects in the two different clusters: if X contains x objects and Y contains y objects, this is the

  mean of the sum of distances (X_i,Y_j) where $X_i \in X, Y_j \in Y$, i = 1...x, j = 1...y:

  $$\text{Average Linkage Distance}(X,Y) = (\Sigma_{i=1...x,\, j=1...y}\, \text{dist}(X_iY_j))\,/\,xy \qquad (18.4)$$

  where *dist* is defined as previously; note that distances of objects to themselves are not counted in this calculation, and neither are symmetric ones on the grounds that the distance from, say Xi to Yj is the same as the distance from Yj to Xi.

There are other linkages, for which see Moisl (2015:209-13). Incidentally, note that dendrograms are more often shown sideways than in the 'icicle' or downward-facing format more familiar to linguists. This is a purely practical matter: an icicle format rapidly broadens out as the number of data objects grows, making it impossible to display on a page.

The main and considerable advantage of hierarchical clustering is that it provides an exhaustive description of the proximity relations among data objects, and thereby more information than a simple partitioning of the data generated by non-hierarchical methods. It has also been extensively and successfully used in numerous applications, and is widely available in software implementations. There are, however, several associated issues.

- How many clusters?
  Relative to a given tree, how many clusters do the data 'really' contain? That is up to the user to decide. Looking at a dendrogram like the one in Fig. 18.15 the answer seems obvious: there are two clusters A and B; each of these itself has some internal cluster structure, but that structure is insignificant compared to the main A/B partition. This intuition is based on the relative lengths of the lines joining subclusters or, equivalently, on the relative values in the joining table: unusually large intervals between successive merges are taken as an indication that the subclusters in question constitute 'main' clusters.

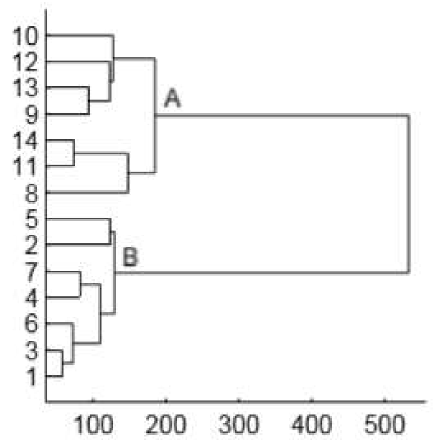


Fig. 18.15 Hierarchical tree showing a clear two-cluster structure

What about a structure like the one in Fig. 18.16, however? There are no obvious main clusters, and depending on the joining level one selects, that is, where one 'cuts' the tree, two, three, four or more clusters can be identified. In Fig. 18.16a the cut is placed so that subclusters below a threshold of 100 are not distinguished, yielding two clusters A and B. In 16b the cut is at 90, yielding three clusters A−C, in 16c there are four clusters A−D for a threshold of 73, and in 16d there are five clusters A−E. Which is the best cut, that is, the one that best captures the cluster structure of the data? There have been attempts to formalise selection of a best cut, but the results have been mixed, and the current position is that the best cut is the one that makes most sense to experts in the subject from which the data comes.

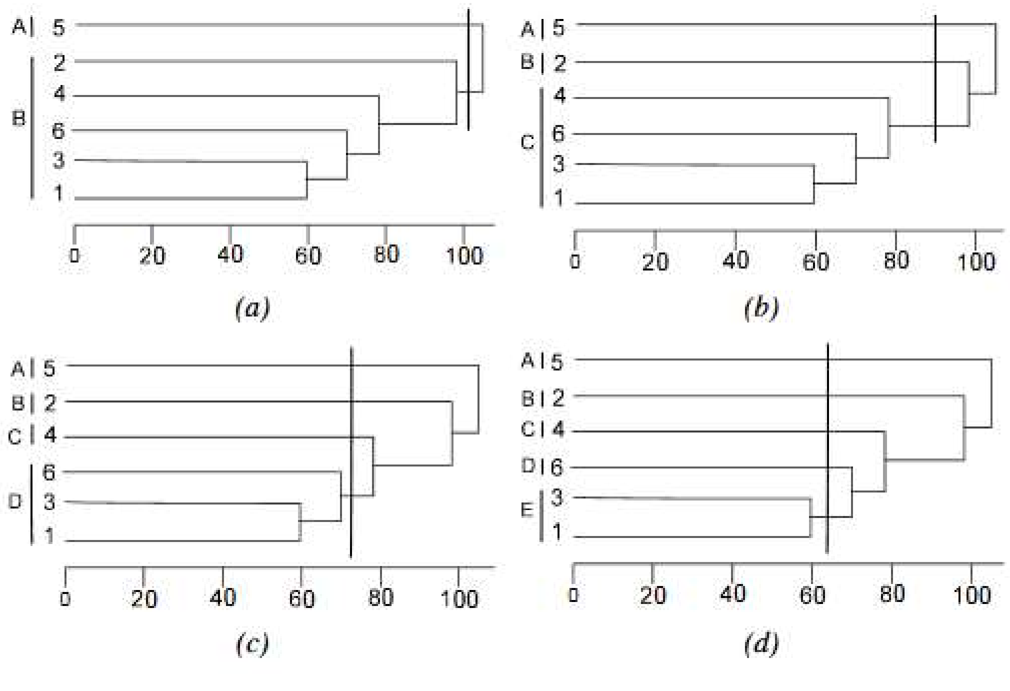


Fig. 18.16 Different 'cuts' of the same dendrogram

- Which tree?
  The clustering literature recognises that different cluster joining criteria can and typically do generate different trees for the same data. For example, Fig. 18.17 shows dendrograms for two different analyses of Table 18.1. Both trees show a two-cluster structure consisting of speakers n01 - n07 at the top of each and the remaining ones below, but the structuring of the latter differs greatly. This is hardly surprising. The various joining criteria articulate different views of how data points should be combined into clusters, and these views find their expression in different cluster trees relative to the same data. It does, however, raise two questions: (i) given several hierarchical analyses of the same data generated by different joining criteria, which analysis should be preferred, and (ii) why? The traditional answer is that an expert in the domain from which the data was taken should select the analysis which seems most reasonable in terms of what s/he knows about the research area. The obvious objection to this is that it is subjective. It runs the risk of reinforcing preconceptions and discounting the unexpected and potentially productive insights which are the prime motivation for use of cluster analysis in hypothesis generation: given a range of different analyses, one might subconsciously look for what one wants to see.

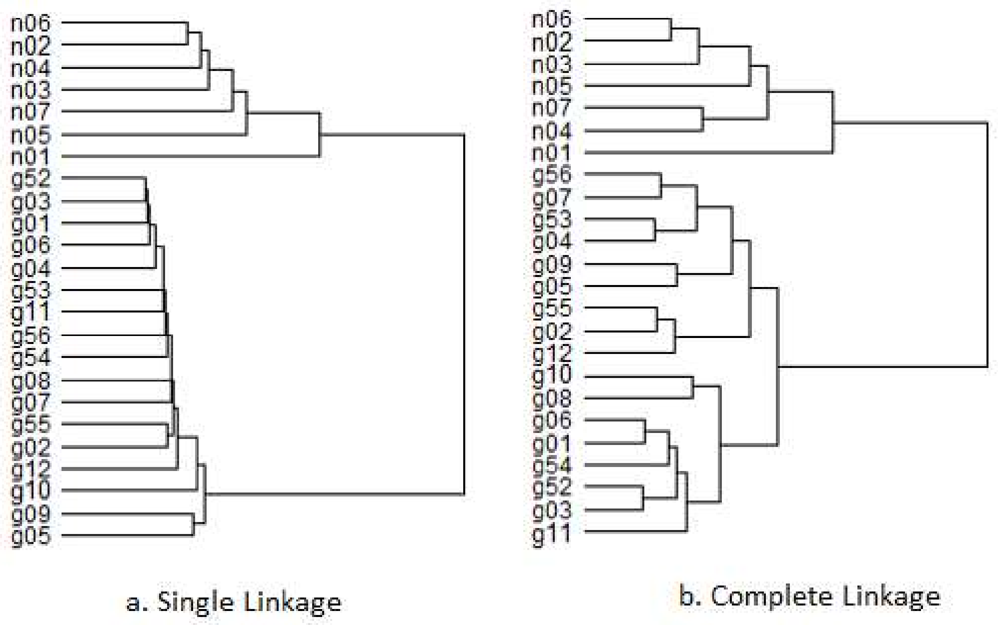


Fig. 18.17 Dendrograms for Table 18.1 generated by different linkage criteria

The aesthetics of tree structuring might also become a selection factor. Looking at the trees in Fig. 18.17, one might find the clear cluster structure of the Complete Linkage tree more appealing than the uninformative chained structure of the Single Linkage one for the g01 – g56 speakers. Ultimately, of course, all human interpretation is subjective, but to be scientifically convincing, subjectivity needs to be constrained as much as possible. One way of constraining tree selection is to observe that there is a fundamental difference between the Single Linkage criterion and the others referred to above, as explained in Moisl (2015:217-21). The ability of Single Linkage to identify a superset of the cluster structures identifiable by the other linkage methods implies that it is more likely than the others correctly to identify the structure latent in any given data, and that it is therefore the most authoritative of the linkage methods: in principle, no matter how aesthetically attractive a non-Single Linkage tree might be, or how much it accords with expert expectation, if it differs substantively from the Single Linkage tree, the suspicion must be that the difference is an artefact of the linkage criterion rather than a reflection of intrinsic cluster structure. In practice there is a caveat, however. Single Linkage does not make a distinction between cluster structure and noise in data, on which more below, and this can generate spurious structures. When the given data is known not to contain much or any noise, Single Linkage is authoritative. Where this is not known, selection of the best cluster tree, that is, the one which best captures the intrinsic cluster structure of the data, must be guided by the various cluster validation methods discussed in Moisl (2015:224-49).

- Outliers and noise
  All the joining criteria are affected by outliers and noise to different degrees and for different reasons, where 'noise' designates a variety of factors such as inaccurate measurement or recording, or accidental or deliberate corruption. Outliers are not a problem for Single Linkage because it simply represents them as one-member clusters distant from the other clusters in the tree; this characteristic in fact makes Single Linkage a good way to check for outliers. It is, however, much affected by noise, which results in chaining like that in Fig. 18.17a; if that chaining is indeed a consequence of noise rather than a reflection of the similarity structure of the speakers' pronunciation, a possible cause is inaccurate transcription. For the other linkage criteria noise is less of a problem, but outliers affect them considerably: because Complete Linkage and the various others are all based in one way or another on the notion of a cluster having a centre of gravity or centroid, an outlier pulls that centre away from its natural location among the other data points, thereby affecting calculation of the centre and consequently distorting the structure of the tree. When using hierarchical cluster analysis, therefore, it is important to identify outliers and to eliminate them or at least to be aware of their presence when interpreting results.

Finally, because of their widespread popularity in data analysis generally, *k-means* and hierarchical clustering continue to be developed in the hope of at least mitigating their problems. For the interested reader, these have to do with linear separability of clusters and the use of nonlinear distance measurement, for which see Moisl (2015:187-201, 217-23).

18.2.3.4 Advanced Topics

Cluster analysis is widely used and highly developed both in terms of the variety of available clustering methods and of theoretical understanding of them (Moisl 2015:Chap.4). Hierarchical cluster analysis and *k-means* are a good place to start, but there is extensive potential for application of other clustering techniques in corpus linguistics. These topics, in particular, are worth pursuing:

- As noted, different clustering methods often give substantively different results. The reason might have to do with noise, or lack of intrinsic cluster structure in the data, or some limitation in the clustering method such as those already referred to, or a predisposition of different methods to find certain shapes of cluster. In such cases, a principled selection is required; cluster validation is a rapidly developing and necessary aspect of cluster analysis (Moisl 2015:224-49).
- Density-based clustering methods can identify a greater range of cluster shapes than

vector space-based ones such as hierarchical and *k-means* (Moisl 2015:192-201).

- Nonlinear distance measurement can be more accurate than linear measures such as Euclidean distance, and can be expected by give more reliable results (Moisl 2015:39-45).

Representative study 1

Moisl, H., Maguire, W., and Allen, W. 2006. Phonetic variation in Tyneside: exploratory multivariate analysis of the Newcastle Electronic Corpus of Tyneside English. In *Language Variation. European Perspectives*, ed. Hinskens, F. Meertens Institute, Amsterdam.

This paper is a sociolinguistic study of phonetic variation among speakers from Tyneside in North-East England.

Research Question Is there systematic phonetic variation in the Tyneside speech community, and, if so, does that variation correlate systematically with social factors?

Data Phonetic data was abstracted from the *Diachronic Electronic Corpus of Tyneside English* (DECTE), a digital corpus of audio-recorded and transcribed speech from Tyneside in North-East England. DECTE is available online and is fully documented at http://research.ncl.ac.uk/decte/, and is described in Corrigan et al. (2012). DECTE includes phonetic transcriptions of 63 audio speaker interviews using a transcription scheme containing 156 phonetic variables. A phonetic usage profile was created for each speaker by counting and recording the number of times the speaker used each of the 156 phonetic segments, and the 63 profiles were assembled into a 63 x 156 data matrix called MDECTE. The MDECTE header contained phonetic segment variable names and corresponding four-digit codes, one per column, and each row has a speaker label. This is exemplified in Table 18.5.

Table 18.5 Schematic of the MDECTE data matrix

	1: δ1 0194	**2:δ2 0198**	**3: Ɔ: 0118**	**...**	**156: Ɛ 0164**
g01	31	28	123	...	0
g02	22	8	124	...	0
...	...	...	...	...	...
n07	19	19	73	...	0

MDECTE was transformed in two ways prior to cluster analysis:

- Because there is substantial variation in the lengths of the individual speaker transcriptions, MDECTE was normalised to compensate for this (Moisl 2015:65-71).
- A majority of the phonetic variables are either very infrequently used or show very little variation across speakers. MDECTE was therefore dimensionality reduced to a 63 x 59 matrix (Moisl 2015:71-92).

Method MDECTE was hierarchically cluster analyzed as shown in Fig. 18.18.

Result The DECTE speakers fell into two clearly defined main clusters, a larger one G containing speakers g01 - g55 and a smaller one N containing speakers n01 - n07, and G itself has well-defined subclusters. When this cluster structure was correlated with social data about the speakers included in DECTE, all the speakers in cluster G were found to be from Gateshead on the south side of the river Tyne, and all those in N were from Newcastle on the north side. Moreover, G.1 contains speakers with somewhat higher levels of education and employment than those in G.2, all of whom had the minimum statutory educational level and were in skilled and unskilled manual employment, and G.2 itself consists of two subclusters for gender, where G.2.2 consists entirely of men and G.2.1 mainly though not exclusively of women. Cluster analysis of MDECTE therefore empirically supports the hypotheses that there is systematic phonetic variation in the Tyneside speech community, and that this variation correlates systematically with social factors.

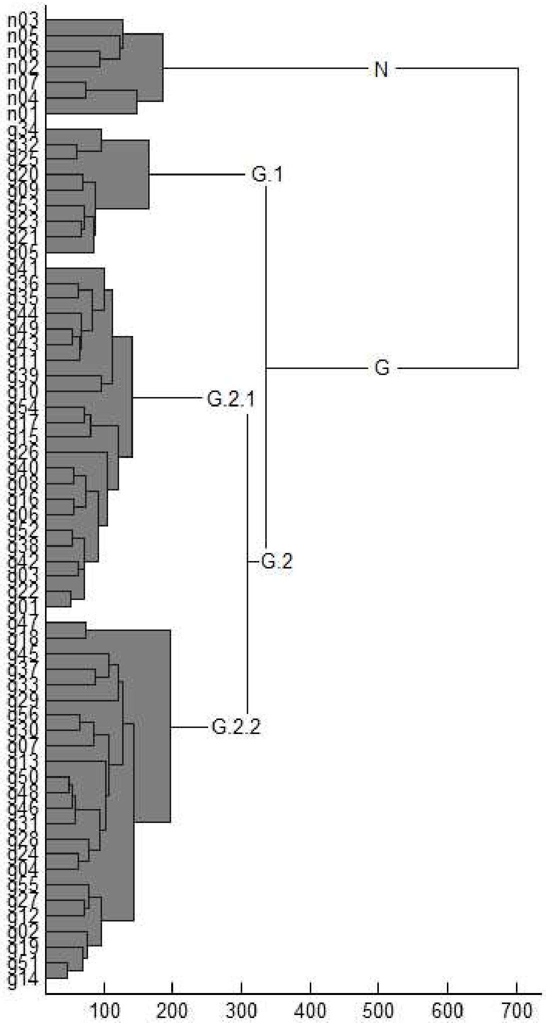


Fig. 18.18 Hierarchical cluster analysis of MDECTE

Representative study 2

Gries, S., and Stefanowitsch, A. 2010. Cluster analysis and the identification of collexeme classes. In *Empirical and Experimental Methods in Cognitive / Functional Research*, eds. Rice, S., and Newman, J., 73-90. CSLI, Stanford CA.

A substantial amount of work has been done on inference of grammatical and semantic lexical categories from text (Korhonen 2010). This work is based on the observation that there are restrictions on which words can co-occur within some degree of contiguity in natural language strings, and that the distribution of words in text can therefore be used to infer grammatical and semantic categories and category relatedness for them; as Firth (1957) put it, "a word is characterised by the company it keeps". This paper addresses the relationship between words and grammatical constructions.

Research Question How can the number and nature of semantic classes that are instantiated by a given set of words be determined without recourse to prior assumptions or intuitions?

Data The usual data creation approach of abstracting and counting the collocates of target words from a corpus was modified by including only the covarying collexemes of target words, that is, words which occur in a defined slot of the same construction as the target word (see Chap. 7).

Method The authors investigated the effectiveness of hierarchical cluster analytic techniques with respect to the research question by determining how well these techniques identify the most prototypical sense(s) of a construction as well as subsenses instantiated by coherent semantic classes of words occurring in it.

Result There was "a relatively coherent classification of verbs into semantic groups" superior to a classification based solely on collocate-counting. The conclusion was that clustering techniques can be a useful step towards making the semantic analysis of grammatical constructions more objective and precise.

18.3 Practical Guide with R

This section contains a guide to *k*-means and hierarchical cluster analysis using R. The MDECTE data matrix, truncated to 32 rows for convenience of exposition, is used as the basis for exemplification.

18.3.1 K-means

The *k*-means analysis described in what follows proceeds in three steps: (i) finding the appropriate number of clusters *k*, (ii) carrying out the analysis for the selected *k*, and (iii) reporting the result.

(i) As noted earlier, one of the ways of establishing *k* is to conduct multiple analyses on the same data using different values of *k* and selecting the one whose SEE is optimal. It was also noted that, because different selections of initial prototypes affect the success with which *k*-means identifies cluster structure, multiple trials with different random initialisations are conducted for each value of *k*, and the one with the smallest SSE is selected. Optimisation of prototypes and the value for *k* selection must, therefore, be integrated: in assessing the optimality of any given *k* in the succession of candidate *k*-values, the SSE for that *k* must itself be optimal. The R code for this follows.

The first three lines assign values to variables used subsequently

```
maxk <- 15
iterations <- 50
lst <- vector()
```

where

- `maxk` is the maximum number of *k*-values to be assessed.
- `iterations` is the number of initial prototypes to be tried for each *k*.
- `lst <- vector()` creates an empty list into which optimal SSE values for the current *k* will be inserted.

The list of optimal SSE values is then constructed.

```
set.seed(123)
for (i in 1:maxk)
 {mk <- kmeans(m, centers=i, iter.max=10, nstart=iterations)
  lst[i] <- mk$tot.withinss}
```

where

- `set.seed(123)` is a technicality. It initialises the R random number generator, which is used to generate random prototype vectors in the code that follows. Specifying a seed ensures that the generator outputs the same sequence each time the `kmeans` function is used, thereby ensuring reproducibility of results for any given data.
- `for (i in 1:maxk)`: for every value in the range 1...`maxk`.
- `mk <- kmeans(m, centers = i, iter.max = 10, nstart = iterations)` uses the R function `kmeans` to calculate `mk`, the *k*-means analysis for any given k in the range 1..`maxk`.

The meanings of the parameters for `kmeans` are as follows:

- `m` is the data matrix.
- `centers = i` is the number of clusters *k* for every successive *k* in the range 1..`maxk`.
- `iter.max = 10` is the number of iterations of the *k*-means algorithm to apply before giving up on the current *k* as unviable.
- `nstart = iterations` is the number of random initial prototypes to try for each *k*. It is important to note that `kmeans` automatically selects and outputs the prototype initialisation with the optimal SSE.
- `lst[i] <- mk$tot.withinss` inserts the optimal SSE for the current k into the SSE list `lst`, and `mk$tot.withinss` indexes the SSE which `kmeans` outputs at each iteration.

The list of SSE values `lst` is now plotted, with the result shown in Fig. 18.19:

```
plot(lst)
```

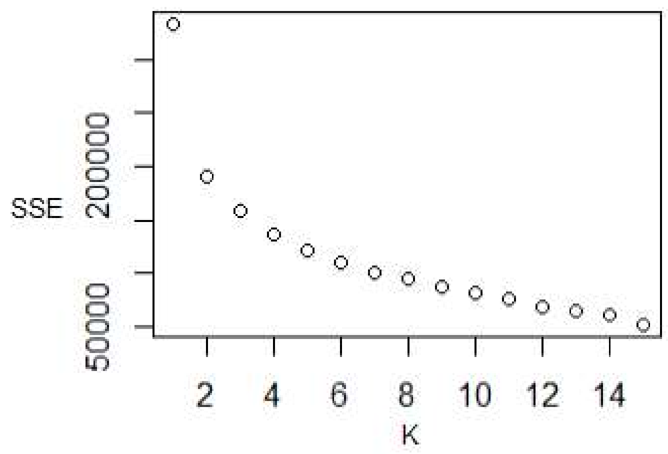


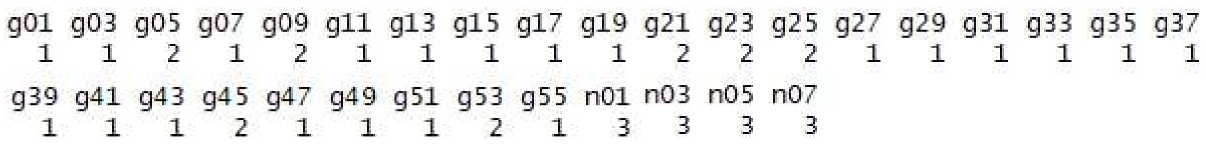
Fig. 18.19 Decrease of SSE for *k* in the range 1..15

Figure 18.19 shows SSE decreasing with increasing *k*. How can this be used to select optimal *k*? The main thing to realise is that optimal *k* is not the one with the smallest SSE: if the range had been extended the SSE would have become ever smaller until it reached 0 when *k* = the number of data objects, at which point every cluster would have had 1 member. Optimality is, instead, the *k* at which the number of clusters is at its most informative, and this is the *k* for which the rate of decrease of SSE starts to level out. Visual inspection indicates that, in Fig. 18.19 the rate of decrease is large up to *k* = 3, and starts to level out thereafter, so the optimal value is taken to be 3. This selection is subjective - it could have been 4 or 5 - but subjectivity is intrinsic to the method.

(ii) The second step invokes the R function `kmeans` again, using the selected *k* via the parameter `centers = 3`.

```
mk <- kmeans(m, centers = 3, iter.max = 10, nstart = 50)
```

(iii) Finally, the variable *mk* contains the result, and includes a range of information whose significance is explained in the R help file for *kmeans*. Most relevant for present purposes is the clustering vector, which assigns each data object to one of the specified three clusters, as shown in Fig. 18.20.



```
g01 g03 g05 g07 g09 g11 g13 g15 g17 g19 g21 g23 g25 g27 g29 g31 g33 g35 g37
  1   1   2   1   2   1   1   1   1   1   2   2   2   1   1   1   1   1   1
g39 g41 g43 g45 g47 g49 g51 g53 g55 n01 n03 n05 n07
  1   1   1   2   1   1   1   2   1   3   3   3   3
```

Fig. 18.20 Clustering vector for *k* = 3

18.3.2 Hierarchical Clustering

Hierarchical analysis is, again, a three-step process: (i) creation of a distance matrix, (ii) clustering, and (iii) presentation of the result.

(i) Generate the Euclidean distance matrix:

```
md <- dist(m)
```

where

- `md` is the distance matrix.
- `dist` is the relevant R distance calculation function.

(ii) Cluster the distance matrix:

```
mdc = hclust(md, method="average")
```

where

- `mdc` is the list output of the hierarchical cluster analysis.
- `hclust` is the R clustering function.
- `method="average"` selects the Average Linkage method; other linkages can be specified.

(iii) Draw the dendrogram to display the clustering (Fig. 18.21). There are numerous graphical options here; a basic one is:

```
plot(mdc, labels=m$V1, hang=-1, cex=0.75)
```

where

- `plot` is self-explanatory.
- `labels=m$v1` tells `plot` that the speaker labels are in column 1.
- `hang=-1` shows all the labels in a uniform horizontal row.
- `cex=0.75` adjusts the size of the label font; this is useful when there are many labels and too large a font makes them overlap and become unreadable.

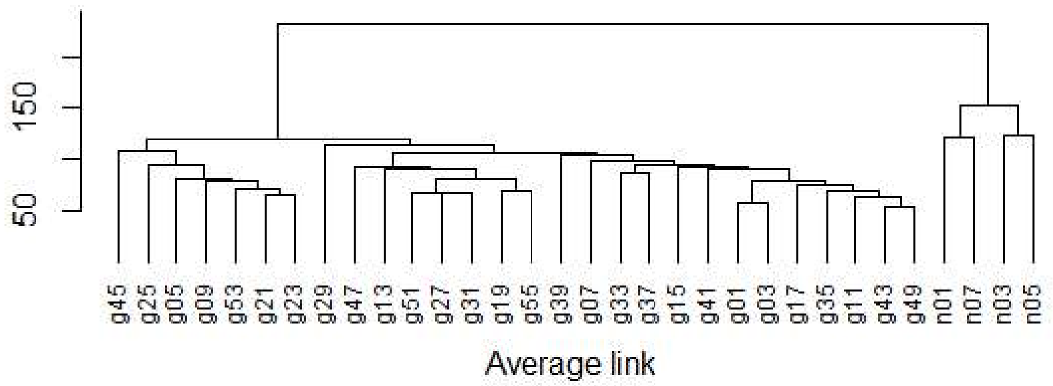


Fig. 18.21 Average Linkage hierarchical clustering of MDECTE

As the number of rows in a data matrix grows, the tree in the 'icicle' format shown in Fig. 18.21 soon runs over the edges of the page. It is more convenient to show it in the arbitrarily-extendible horizontal format as used in Fig. 18.18 for example. To do this and at the same time retain the font-size adjustment facility, it is necessary to extend the native R clustering graphics with a graphics package, one of which is `dendextend`.

- Convert the tree generated by `hclust` to an R *dendrogram* object:

```
mdcd <- as.dendrogram(mdc)
```

- Set the font size:

```
mdcd <- set(mdcd, "labels_cex", 0.75)
```

- Plot the dendrogram (Fig. 18.22):

```
plot_horiz.dendrogram(mdcd)
```

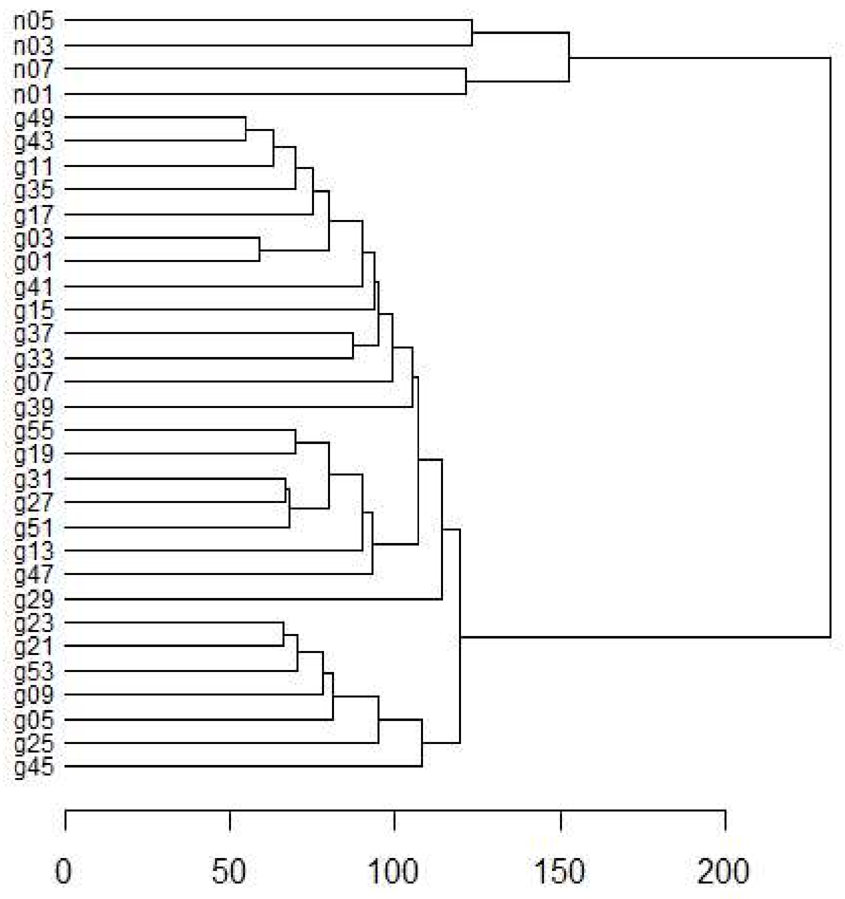


Fig. 18.22 Fig. 18.18 shown horizontally

There are several packages in addition to `dendextend` which offer an extensive range of graphics options, including `ggdendro` and `APE: Analyses of Phylogenetics and Evolution in R language`.

18.3.3 Reporting Results

K-means and hierarchical clustering are mathematical, not statistical, procedures. As such, there are no statistical measures such as standard deviation or p-value to report when preparing results for publication. This does not, however, mean that it is sufficient to present nothing more than a clustering vector or a dendrogram. It is important to provide the reader with as much information as possible about the framework in which the analysis was conducted.

- *Data*: How was the data collected? How large is the data set? What does the data matrix look like (including an example)? How, if at all, was the data transformed, for example by removal of outliers, or by variable scaling, or by normalisation? See Chap. 26 for more specific guidelines on data description.
- *Clustering method*: As noted, different hierarchical methods can give different results. Simply presenting the one that, for whatever reason, the researcher finds most appealing is insufficient. Instead, several methods should be applied to the data, agreements and disagreements among them identified, and reasons for any disparities investigated. Hierarchical results should, moreover, be corroborated via *k*-means and / or a selection of the many other available clustering methods. Finally, validation is increasingly being regarded as essential in reporting of clustering results (Moisl 2015:224-49).

Further Reading

The literature on clustering is vast and, in general, quite technical, so unless one is mathematically sophisticated it is best to work up the scale of difficulty gradually.

Divjak, D., and Fieller, N. 2014. Clustering linguistic data. In *Corpus Methods for Semantics*, eds. Glynn, D., and Robinson, J., 405-441. John Benjamins, Amsterdam.

Divjak & Fieller's chapter is the best current follow up to the present chapter. It is aimed at about the same level as the present one, and the two complement one another well by offering independent views and emphases. Thereafter:

Moisl, H. 2015. *Cluster Analysis for Corpus Linguistics*. de Gruyter, Berlin.

This book-length account not only offers coverage of a substantially greater range of methods than that offered by Divjak & Fieller and by the present chapter, but also includes extensive discussion of data issues and cluster validation as well as a detailed overview of cluster analysis applied in various linguistics sub-fields.

References

Corrigan, K., Mearns, A., and Moisl, H. 2012. *The Diachronic Electronic Corpus of Tyneside English*. http://research.ncl.ac.uk/decte/. Accessed 12 June 2019.

Delmestri, A. and Cristianini, N. 2012. Linguistic phylogenetic inference by PAM-like matrices. *Journal of Quantitative Linguistics* 19: 95– 120.

Deza, M., and Deza, E. 2009. *Encyclopedia of Distances*. Springer, Berlin.

Firth, J. 1957. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*, ed. Firth, J. Blackwell, Hoboken NJ.

Gries, S. 2010a. Corpus linguistics and theoretical linguistics: a love-hate relationship? Not necessarily... . *International Journal of Corpus Linguistics* 15: 327–343.

Gries, S., and Stefanowitsch, A. 2010b. Cluster analysis and the identification of collexeme classes. In *Empirical and Experimental Methods in Cognitive / Functional Research*, eds. Rice, S., Newman, J., 73-90. CSLI, Stanford CA.

Grieve, J., Speelman, D., and Geeraerts, D. 2011. A statistical method for the identification and aggregation of regional linguistic variation. *Language Variation and Change* 23: 193–221.

Hauer, B. and Kondrak, G. 2011. Clustering Semantically Equivalent Words into Cognate Sets in Multilingual Lists. In *The 5th International Joint Conference on Natural Language Processing (IJCNLP 2011),* 865 – 873.

Kessler, B. 2008. The mathematical assessment of long-range linguistic relationships. *Language and Linguistics Compass* 2: 821–839.

Köhler, R. and Hoffmann, C. 1995. *Bibliography of Quantitative Linguistics*. John Benjamins, Amsterdam.

Korhonen, A. 2010. Automatic lexical classification: bridging research and practice. *Philosophical Transactions of the Royal Society A* 368: 3621-3632.

Meschenmoser, D. and Pröll, S. 2012. Fuzzy clustering of dialect maps using the empirical covariance. Grouping maps in dialect corpora based on spatial similarities. *International Journal of Corpus Linguistics* 17: 176–97.

Mirkin, B. 2011. *Core Concepts in Data Analysis: Summarization, Correlation, and Visualization*. Springer, Berlin.

Moisl, H., Maguire, W., and Allen, W. 2006. Phonetic variation in Tyneside: exploratory multivariate analysis of the Newcastle Electronic Corpus of Tyneside English. In *Language Variation. European Perspectives.* Ed. Hinskens, F. Meertens Institute, Amsterdam.

Moisl, H. 2015. *Cluster Analysis for Corpus Linguistics*. de Gruyter, Berlin.

Ruette, T., Speelman, D., and Geeraerts, D. 2013. Measuring the lexical distance between registers in national variaties of Dutch. In: *Pluricentric Languages: Linguistic Variation and Sociocognitive Dimensions.* Ed. Soares da Silva, A. Mouton de Gruyter, Berlin. 541–554.

Wieling, M., Nerbonne, J., and Baayen, H. 2011. Quantitative Social Dialectology: Explaining Linguistic Variation Geographically and Socially. *PLoS ONE* 6; nr. 9.

Xu, R., and Wunsch, D. 2009. *Clustering*. Wiley, Hoboken NJ.