Cluster analysis
Finding structure in linguistic data

Dagmar Divjak and Nick Fieller
University of Sheffield

Cluster analysis is an exploratory data analysis technique, encompassing a number of different algorithms and methods for sorting objects into groups. Cluster analysis requires the analyst to make choices about dissimilarity measures, grouping algorithms, etc., and these choices are difficult to make without an understanding of their theoretical implications and a very good understanding of the data. This chapter provides an introduction to the distance measures and clustering algorithms most commonly used for cluster analytic work. Different from Baayen (2008), Johnson (2008) and Gries (2009), its main aim is to equip the researcher with at least a basic understanding of what is happening when a dataset is explored with the help of a particular cluster analytic technique.

Keywords: clustering algorithms, distance measures

1. Introduction

> We organisms are sensorimotor systems. The things in the world come in contact with our sensory surfaces, and we interact with them based on what that sensorimotor contact "affords". (…) At bottom, all of our categories consist in ways we behave differently toward different kinds of things, whether it be the things we do or don't eat, mate with, or flee from, or the things that we describe, through our language, as prime numbers, affordances, absolute discriminables, or truths. And isn't that all that cognition is for – and about?
>
> (Stevan Harnad 2005: *To Cognize is to categorize: Cognition is categorization*)

One of the key concepts in cognitive linguistics is categorization. To be able to categorize things is a necessary and innate capacity: we need to be able to recognize, distinguish and understand in order to survive. Our categories signal, for example, whether the mushroom we pick is edible or not, and whether the animal we encounter is harmless or dangerous. Survival is the main goal of cognition. Categorization is equally fundamental in language. Growing up, we not only learn which categories are relevant for us to function in our environment. We also acquire the categories of our language and learn to use a limited number of words and rules to name a large number of different items and to express an unlimited number of experiences.

And things do not stop with language. As is the case in other disciplines, categorization is also important in the scientific study of language. As early as the 5th century BC, Sanskrit grammarians grouped words into classes – that would later become known as parts of speech – distinguishing between inflected nouns and verbs and uninflected pre-verbs and particles. Categorization efforts have been carried out across linguistic sub-disciplines, ranging from phonology, morphology, syntax and semantics, to discourse analysis and pragmatics. Despite its long history, even for parts of speech there is currently no generally agreed-upon classification scheme that would apply to all languages, or even a set of criteria upon which such a scheme should be based. In some cases, linguists are only now trying to organize entities into groups so that they can be compared and described.

With desktop computers replacing filing cabinets, it has become easy for linguists to create very large databases that contain information on a multitude of properties. At the same time, this complexity may make it too difficult for the human analyst to detect any structure. One way of solving this issue is by running a cluster analysis. Cluster analysis (a term first used by Tryon in 1939) is a multivariate analysis technique that organizes information about how similar objects or entities are so that groups, or clusters, can be formed. Pioneered in machine learning, cluster analysis has found its way via sociolinguistics (e.g. Shaw's 1974 work on dialectal boundaries) to linguistics, where it has now been used to describe a wide range of linguistic phenomena (see references elsewhere in this volume). It is also reaching beyond linguistics into other Arts and Humanities disciplines, aiding, for example, in the classification of texts according to relevant dimensions (see Alviar 2008 for the application of cluster analysis in the style characterization of New Testament texts).

Cluster analysis is an exploratory data analysis technique, encompassing a number of different algorithms and methods for sorting different objects into groups in such a way that the similarity between two objects of the same group is maximal and the similarity between two objects that belong to different groups is minimal. In other words, cluster analysis can be used to discover structures in data and it does this without explaining why that structure exists. Cluster analysis is thus not a routine single statistical test based on probability theory; instead, it is a data analytic technique, a collection of different algorithms that put objects into clusters according to well-defined similarity rules. It is mostly used when we do not have any *a priori* hypotheses, but are in the exploratory phase of our research. Yet in contrast to other exploratory methods of this type such as Principal Components Analysis and Multidimensional scaling, cluster analysis requires the analyst to make choices, about dissimilarity measures, grouping algorithms, etc., and these choices are difficult to make without having an understanding of their theoretical implications and a very good understanding of the data. This latter requirement is particularly important, since in contrast to many other statistical methods, there seem to be fewer diagnostics informing of the weaknesses of any classification solution proposed.

Before getting started, it is important to stress the focus of this chapter. Like Baayen (2008: Section 5.1.5), Johnson (2008: Ch. 6) and Gries (2009: Ch. 5.5) before us, this chapter provides an introduction to the distance measures and clustering algorithms most commonly used (for an idea of the amazing variety of functions for cluster analysis that R has to offer, see http://cran.r-project.org/web/views/Cluster.html). Our main aim is, however, to equip the researcher with at least a basic understanding of what is happening behind the scenes when s/he explores his/her data with the help of a particular cluster analytic technique using R (without actually answering the one unanswerable question: "which method of clustering is best?"). Slightly more technical details may not suit all readers on first reading. To avoid letting linguistic opinions overshadow statistical decisions, we have opted to illustrate our chapter with a topic that would hardly raise any interest in linguistic circles, i.e. the relatedness of languages, based on the words they use to express the numbers 1 through 10.

Cluster analysis only works on data that is "prepared" in some way. In general, this preparation equals "operationalizing" parameters deemed relevant in the description of the data, an issue we will elaborate on in Section 2.1. Although cluster analysis is relatively simple and can use a variety of input data, many aspects rely on *ad hoc* and intuitive justification: only certain particular aspects of the techniques are supported by established statistical theory. Therefore, most of the guidelines for using cluster analysis are rules of thumb. Once decisions have been made on key steps of the cluster analysis, topics we will go into in Sections 2.2 and 2.3, the cluster analysis technique provides a degree of objectivity in determining clusters or groups of similar entities. This has the advantage of identifying which elements of the analysis are subjective and then specifying precisely what follows from these subjective decisions, as we will do in Sections 2.4 and 2.5.[1]

1. This allows two types of comparisons. On the one hand, comparisons can be made between analyses based on different subjective decisions applied to the same entities; on the other hand, comparisons can be made between structures of different sets of entities based on matching subjective inputs. In other words, two cluster analyses cannot be compared/contrasted unless they have been applied to the same set of objects or have used the same similarity measure and clustering algorithm.

2. Steps in conducting a cluster analysis

There are several basic cluster analysis steps:

– data collection and selection of the variables for analysis (Section 2.1)
– generation of a similarity matrix and clustering algorithm (Sections 2.2 and 2.3)
– decision about the number of clusters and their interpretation (Section 2.4)
– validation of the cluster solution (Section 2.5)

These steps are interrelated. The type of variable selected for analysis will impose restrictions on the choice of how to measure similarities, and the choice of clustering algorithm will influence the interpretation of the clusters detected.

2.1 Coding and standardization of variables

An important contribution to the statistical analysis of linguistic data, of any data really, is made by the variables used to capture the phenomenon. A common myth is that corpus linguists do everything automatically, which would make corpus linguistic techniques unsuited for the study of meaning. Yet at the heart of the corpus-based study of linguistic phenomena is the manual annotation of examples. This requires the analyst to read and analyze the examples one by one, just as s/he would do with a list of examples collected in a notebook or written out on cards. One of the main differences between a corpus linguist and other linguists is that the corpus linguist selects a random sample from a representative and balanced collection of texts that represent one or more varieties of the language s/he is studying. The corpus linguist then works with that sample and is not allowed to include additional sentences that would nicely illustrate his/her point, nor to remove sentences that disprove his/her account.

An additional challenge the corpus linguist faces is that s/he needs to operationalize linguistic parameters in such a way that they can be applied consistently to a large number of examples and can be fed into a computer for statistical analysis. In other words, the corpus linguist has to decide how to put his/her variables onto a numeric scale of some sort, in a way that the variables and variable levels cover all instances encountered in the examples. This typically implies an initial analysis of the data, since not everything is quantified naturally.

Let us consider, as an example, the popular linguistic issue of discovering family relations between languages. Given that cardinal numerals seem to only slowly change their form, and certainly keep their meaning, they testify to older stages of the language. For this particular example, we limit ourselves to cardinal numerals from 1 to 10 in some Romance, Germanic and Slavonic languages (see Tables 1, 2 and 3).[2]

2. The study was inspired by a study looking at numerals from one to ten in 11 languages, published in Wichern and Johnson (2007). The larger dataset, comprising 51 languages, was collected by Nick Fieller from phrasebooks and exchange students.

Table 1. Numerals in some Romance languages

English	French	Occitan	Catalan	Spanish	Asturian	Galician	Portugese	Italian	Romanian
one	un	un	un	uno	ún	um	um	uno	unu
two	deux	dos	dos	dos	dos	dous	dois	due	doi
three	trois	tres	tres	tres	tres	três	três	tre	trei
four	quatre	quatre	quatre	cuatro	cuatro	catro	quatro	quattro	patru
five	cinq	cinc	cinc	cinco	cinco	cinco	cinco	cinque	cinci
six	six	sièis	sis	seis	seyes	ceis	seis	sei	sase
seven	sept	sèt	set	siete	siete	sete	sete	sette	sapte
eight	huit	uèch	vuit	ocho	ocho	oito	oito	otto	opt
nine	neuf	nòu	nou	nueve	nueve	nove	nove	nove	noŭa
ten	dix	dètz	deu	diez	diez	dez	dez	dieci	zece

Table 2. Numerals in some Germanic languages

English	Dutch	German	Frisian	Norwegian	Danish	Swedish	Icelandic
one	een	ein	ien	en	en	ett	einn
two	twee	zwei	twa	to	to	två	tveir
three	drie	drei	trije	tre	tre	tre	Þrír
four	vier	vier	fjouwer	fire	fire	fyra	fjórir
five	vijf	fünf	fiif	fem	fem	fem	fimm
six	zes	sechs	seis	seks	seks	sex	sex
seven	zeven	sieben	sân	sju	syv	sju	sjö
eight	acht	acht	acht	atte	otte	åtta	átta
nine	negen	neun	njoggen	ni	ni	nio	níu
ten	tien	zehn	tsien	ti	ti	tio	tíu

If we measure the similarity between number words, could we use that information to deduce family relationships? And how can we do that? Which variables would we use? Variables can be any of several different types. Recognizing which type a variable belongs to is important because this affects what choices of analyses are open to the investigator at a later stage. The initial distinction is between categorical, ordinal and numerical variables. Within these, various levels of refinement are possible.

A simple, and certainly imperfect idea, yet one that yields a lot of useful information, is to use current spelling in terms of initial letters (pronunciation specifics could be added, as we will demonstrate in Section 2.5); other measures, e.g. number

Table 3. Numerals in some Slavonic languages

English	Ukranian	Russian	Bulgarian	Macedonian	Serbo-Croat	Bosnian	Slovene	Czech	Polish	Slovak
one	один	один	едно	един	jedan	jedan	ena	jedna	jeden	jeden
two	два	два	две	два	dva	dva	dve	dvě	dwa	dva
three	три	три	три	три	tri	tri	tri	tři	trzy	tri
four	чотыре	четыре	четири	четири	četiri	cxetiri	štiri	čtyři	cztery	štyri
five	пять	пять	пет	пет	pet	pet	pět	pět	pięć	päť
six	шістъ	шесть	шест	шест	šest	sxes	šest	šest	sześć	sesť
seven	сімъ	семь	седем	седум	sedam	sedam	sedem	sedm	siedem	sedem
eight	вісім	восемь	осем	осум	osam	osam	osem	ost	osiem	osen
nine	девятъ	девять	девет	девет	devet	devet	devet	devět	dziewięć	deväť
ten	десятъ	десять	десет	десет	deset	deset	deset	deset	dziesięć	desäť

of syllables or last letters, output nonsense. If the spelling of the first letter is our variable of interest, we would have over 30 different variable levels that lack any intrinsic ordering, making it a classical example of a categorical variable (sometimes called a nominal variable), i.e. a variable that has two or more categories, but there is no intrinsic ordering to the categories. There are plenty of categorical variables in linguistics. Think of, for example, masculine/feminine/neuter gender or imperfective/perfective aspect. There is no intrinsic ordering to these categories, no agreed way to order these from highest to lowest or from smallest to largest, etc. If the variable has a clear ordering, then that variable would be an ordinal variable, as described below. Categorical variables do not have numeric values so it is not possible to perform arithmetic calculations (such as calculation of averages) on them either. A categorical variable with two categories (i.e. a *binary variable*) may be coded as taking values 0 and 1 but this is merely convention. Even if male is coded as 0 and female as 1, it still makes no sense to calculate an average gender.

Ordinal variables are similar to categorical variables. The difference between the two is that ordinal variables do show a clear ordering. Ordinal linguistic variables do not spring to mind as easily as categorical variables do, probably because they require some form of data analysis. Let us consider the following example. Suppose you have a variable that encodes the level of control you have over an action, with three categories, for example "low" for verbs like *forget*, "medium" for actions like *find* and "high" for verbs like *copy*. In addition to being able to classify control-situations into these three categories, you can order the categories as low, medium and high. Although we can order situations according to the level of control we have over them, the spacing, i.e. the size of the difference, between the levels may well be inconsistent. The difference between medium and high control may well be much bigger than the difference between low and medium control. As with categorical variables it is not possible to perform arithmetic calculations on ordinal variables unless they are given numerical scores to reflect the ordering. However, assigning values 1, 2, 3, … will presume that the difference between the first two levels is the same as that between the second and third. Whether this is appropriate is a subjective decision based on the expertise of the linguist. Often, assigning scores to ordinal variables (i.e. converting them to numerical variables) is the only way of taking the ordering into proper account when analysing ordinal data. An alternative would be to ignore the ordering and treat it as a mere categorical variable.

Numerical variables are the easiest to handle in statistics. A distinction is sometimes made between *interval* and *ratio* numerical variables. Interval variables are measured on a scale that does not have an unambiguous zero point (such as temperatures) while ratio variables are on a scale with a clear zero. This distinction is rarely important but may matter: e.g. saying the temperature today is twice as warm as yesterday is nonsensical since it depends on the scale used (^{0}F or ^{0}C), while an object is always twice as long as another whatever units are employed. However, it is possible to say that the difference between two values of an interval variable (e.g. temperature) is twice that between two others; this makes many common arithmetic operations possible on interval variables. Numerical variables may take only *discrete* values (e.g. integers) or they may be *continuous*, i.e. take any value within a range recorded to as many decimal values as desired. Variables may be restricted to taking only positive values (e.g. counts or frequencies or percentages) or they might be able to take both positive and negative values, e.g. ones derived as differences between other measures.[3]

2.2 Calculation of distances (similarities) between objects

Let us return to our example in which we have decided to use the spelling of the first letter of the cardinal numeral as the variable of interest. In order to decide which languages are more similar, we need more than just a variable to explore the cardinal numerals; we also need a way to assess similarity or dissimilarity between the numerals. In the context of cluster analysis, the term *similarity* has its everyday meaning of how similar entities are. More specifically, *similarity* is a numerical measure of how similar

3. To illustrate the restrictions imposed by the type of variable on choices available at later stages of the analysis it is useful to consider which familiar summary statistics can be calculated for the various types. For categorical variables only the mode (or the most commonly occurring value) can be used. For ordinal variables the median can also be used as a descriptive statistic. For any numerical variable, whether interval or ratio, all the commonly used statistics such as mean, variance, standard deviation, analysis of variance, regression etc. can be used but it is only for ratio variables that it is possible to use coefficients of variations or geometric mean or logarithms.

entities are: the larger that number, the more similar they are. The term *distance* or *dissimilarity* is the converse, i.e. how unlike the entities are. Here, objects are more dissimilar if the numerical measure is larger. In some situations it is more natural to think of similarities between objects, e.g. we could regard two words as being similar if they have similar meanings. In a totally different context we might think it more natural to concentrate on dissimilarities or distances, e.g. difference in percentages of occurrence of two words in a given corpus.

Commonly, similarities are based upon a combination of several properties or variables. For example, we might want a similarity between words to be based not just on meaning but on frequency of usage, number of syllables, overlap of phonemes or almost any conceivable property of the word, provided we could put that property onto a numeric scale of some sort. The numeric scale might be very coarse, e.g. 0, 1, 2 or 3 corresponding respectively to *not at all*, *slightly*, *fairly* and *very similar* in meaning (all subjective judgements, note), or it might be a more continuous measure with many possible numeric values, such as a percentage.

Let us look at the Germanic numerals. As we can see in Table 4, if we take the single first letter at face value, the numeral *one* begins with *e* in virtually all Germanic languages listed, except Frisian and English. *Two* is spelled with a *t* in all languages except German. The first letter of *nine* is *n* in all languages, and so on. We could decide to indicate identity in spelling with 1 and non-identity or difference with 0 for each of the numerals. Given that we are looking at 10 numerals in each language, we can sum 1's (same) and 0's (different) to get overall a similarity/dissimilarity score out of ten per language-pair. On this count, the similarity between Dutch and German, two neighbouring languages, is 5 out of 10, since 5 numerals share an identical first letter, scoring 1 each, and 5 do not, scoring 0 each. On a similar count, the similarity between Swedish and Danish is 9 out of 10, with only the numeral 8 having a differently spelled first letter.

As mentioned before, similarities and dissimilarities are closely related. A distance between a pair of towns also reveals how close they are. Numerically it is easy

Table 4. A dissimilarity matrix for the spelling of the first letter in Germanic numerals from 1 to 10

	English	**Dutch**	**German**	**Frisian**	**Norwegian**	**Danish**	**Swedish**
English	0	7	5	2	2	2	2
Dutch	7	0	4	6	5	6	5
German	5	4	0	4	3	4	3
Frisian	2	6	4	0	1	2	1
Norwegian	2	5	3	1	0	1	0
Danish	2	6	4	2	1	0	1
Swedish	2	5	3	1	0	1	0
Icelandic	2	5	3	1	0	1	0

to construct a similarity from a dissimilarity. For example, a percentage of agreement can be converted to a percentage of disagreement by subtraction from 100%. The 0, 1, 2, 3 scale coding for *not at all*, *slightly*, *fairly* and *very similar* referred to above can be converted to a measure of dissimilarity by subtracting the ranks from 3. This yields 0, 1, 2, 3 indicating *very*, *fairly*, *slightly* and *not at all dissimilar*. The reason for emphasising that similarities and dissimilarities can be calculated from each other is that, although it may be more natural to think of a similarity between objects in many situations, most computer packages such as R will require a list of dissimilarities, rather than similarities.

In our example, we can subtract similarities from 10. The dissimilarity between Swedish and Danish, for example, is 1, since only the numeral 8 has a differently spelled first letter, the nine remaining numerals being identical in the spelling of the first letter. If we collect the dissimilarities for all language pairs, we obtain a dissimilarity matrix (Table 4).

The way in which the similarity or distance between objects is calculated will determine how objects are grouped together by a computer algorithm into a system of clusters, sub-clusters, subsubclusters, etc. In particular, if the measure of similarity depends upon severable variables or properties it matters crucially what relative weights are given to the various properties. For example, we might regard frequency of usage as being twice as important as the number of syllables in reflecting the semantic similarity between words, so we would want the variable 'usage' to be given twice the influence or weight of 'number of syllables' in an overall measure of similarity. Of course, this is a subjective decision based upon expert knowledge. In fact, giving equal weight to all properties is also a subjective decision even though it might look as if an attempt at objectivity has been made.

The sections below describe a few ways of combining properties or variables into measures of similarity or dissimilarity (or distance). In most cases, it is assumed that the variables are given equal weight (i.e. are regarded as equally important) but this is for convenience and it should not be overlooked that using equal weightings implies a specific decision of equal importance. Which of the various measures is "best" in any given situation is again a subjective decision but some restrictions on which measures can be used are imposed by the type of data available, e.g. whether the variables are categorical or whether they are continuous ratio data or whether there are some of each type; we will take this up in some detail below.

2.2.1 *Categorical variables*

Some measures of similarity for categorical variables are designed specifically for binary categorical variables. At first sight this may appear to be a severe restriction, but in principle it is always possible to define a set of "dummy" binary variables to indicate which category an object has. Most recent computer packages handle this step, if it is required, internally and automatically so that the process is hidden from the user; therefore, we will not describe it here. Furthermore, there is a drawback to converting categorical variables with large numbers of categories into dummy binary ones, and so it is generally better to use measures designed specifically for multi-level categorical variables.

For categorical variables we focus on two measures, the application of which is not limited to nominal variables, i.e. *Matching coefficients for binary variables* and the *Jacard coefficient*. After a brief description of their mathematical properties, we will illustrate how this works on a simple example.

Matching coefficients for binary variables: If all the variables measured in an individual observation are binary, then the *simple matching coefficient* is the total number of matches (of both 1s and 0s) divided by the number of variables. This gives equal weight to positive and negative matches. This is reasonable for variables such as gender with 1 coding for male and 0 for female or where 1 and 0 code for presence and absence and a co-absence is of some significance, but it might be misleading in others.

In cases where co-absences are non-informative, a modification of the simple matching coefficient is the *Jacard coefficient* obtained by ignoring those variables with negative matches in the calculation, i.e., it is calculated from the total number of positive matches divided by the number of variables where at least one of the two individuals has value 1 for that variable. If all variables match negatively then the similarity is conventionally taken to be 0.

Apart from these two basic matching coefficients, other modifications have been suggested which affect the weights given to variables where only one of a pair of binary variables is present for a particular individual. Some of these are discussed in detail by Gower and Legendre (1986).

To illustrate the calculation of the *Matching Coefficients* with a simple example, consider Table 5 which gives the values of two categorical variables on six languages.

Here, the two variables are *Family* (with possible values *Germanic* and *Slavic*) and *Alphabet* (with possible values *Latin* and *Cyrillic*). These variables can be represented as binary variables by coding *Germanic* as 1 and *Slavic* as 0, and *Latin* as 1 and *Cyrillic* as 0, giving the result shown in Table 6. The choice of which values are coded as 1 and which as 0 is arbitrary here and will not affect the result. This may not always be

Table 5. An example of two categorical variables measured on six languages

Language	**Family**	**Alphabet**
English	Germanic	Latin
German	Germanic	Latin
Dutch	Germanic	Latin
Russian	Slavic	Cyrillic
Polish	Slavic	Latin
Serbian	Slavic	Cyrillic

Table 6. An example of two binary variables measured on six languages

Language	Family	Alphabet
English	1	1
German	1	1
Dutch	1	1
Russian	0	0
Polish	0	1
Serbian	0	0

Table 7. Table of similarities between languages based on two binary variables with a simple matching coefficient

	English	German	Dutch	Russian	Polish	Serbian
English	1	1	1	0	0.5	0
German	1	1	1	0	0.5	0
Dutch	1	1	1	0	0.5	0
Russian	0	0	0	1	0.5	1
Polish	0.5	0.5	0.5	0.5	1	0.5
Serbian	0	0	0	1	0.5	1

the case, for example, where 1 indicates presence and 0 absence of some property of attribute. Whether co-absence of a property carries the same significance as co-presence is not obvious and can only be decided within the context of the example.

The value of the simple matching coefficient used to measure the similarity between English and German is (1+1) / 2 = 1: the two languages match for the variables *Family* and *Alphabet* and this sum is then divided by 2 because there are two variables in total. The similarity between English and Polish is (0+1) / 2 = 0.5 since there is no match on the first variable but there is on the second. Between English and Russian the similarity is 0 because there are no matches on either of the variables. The complete table of similarities is given in Table 7.

Note that Table 7 is symmetrical about the main diagonal: the similarity between Russian and Polish, for example, is the same as that between Polish and Russian.

The calculations above have naturally led to measures of *similarity*. There are, however, occasions where it is more convenient to work with measures of *dissimilarity*, not least when computer packages such as R require it for entry into further automatic statistical analysis. The measure of similarity used is necessarily between 0 and 1 (since it is scaled by the total number of possible matches) with 1 indicating maximum similarity. This makes it easy to convert the values to a measure of dissimilarity by subtracting them from 1, giving values where 0 indicates minimum dissimilarity and 1 maximum dissimilarity. The result is shown in Table 8.

Table 8. Table of dissimilarities between languages based on two binary variables with the complement of a simple matching coefficient

	English	German	Dutch	Russian	Polish	Serbian
English	0	0	0	1	0.5	1
German	0	0	0	1	0.5	1
Dutch	0	0	0	1	0.5	1
Russian	1	1	1	0	0.5	0
Polish	0.5	0.5	0.5	0.5	0	0.5
Serbian	1	1	1	0	0.5	0

Table 9. Table of dissimilarities between languages based on two binary variables with the complement of a Jacard matching coefficient

	English	German	Dutch	Russian	Polish	Serbian
English	0	0	0	1	0.5	1
German	0	0	0	1	0.5	1
Dutch	0	0	0	1	0.5	1
Russian	1	1	1	0	0	0
Polish	0.5	0.5	0.5	0	0	0
Serbian	1	1	1	0	0	0

If we ignore "negative" matches, i.e. a match between zeroes as zeroes indicate absence of similarity here, then the only changes in Table 7, derived from the simple matching coefficient, involve pairs where at least one match (but not all) is negative; in this case this affects Russian and Polish and Serbian and Polish. The exclusion of cases where all variables match "negatively" is justified because in those cases the distance is maximal and conventionally taken as 1. The results are shown in Table 9.

2.2.2 *Numerical variables*

For numerical variables, whether interval or ratio, it is usual to define distances, since most of the definitions are easily described in geometric terms. It is presumed that all the variables measured on an object are in the same units. This might mean that they are all percentages or all lengths measured in centimetres or weights in grams, or it could mean that they are standardised in some other way (for numerical variables this could be done by dividing by their standard deviations, and for categorical variables by dividing by their range, for example). Although we have described special coefficients for binary variables, practical experience indicates that the various measures described below can also be used for collections of binary variables provided that there are more than ten to twenty binary variables. The results of the analyses seem interpretable and are supported by other available evidence. Some basic and widely-applied methods include:

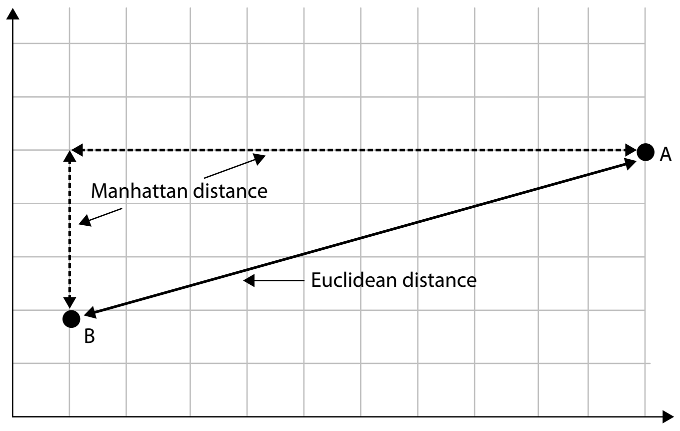

Figure 1. The Euclidean and Manhattan distances

Euclidean distance. The ordinary geometric distance between two points, A and B, marked on a piece of paper, as illustrated in Figure 1, is the Euclidean distance.

With only two variables it is possible to represent all the individuals in a collection by points whose coordinates are given by the values. The Euclidean distance between two points is then obtained by summing the squared differences between the pairs of corresponding values for the two individuals and taking the square root of the sum. When there are several variables the procedure is the same except that the sum is over the squared differences of corresponding values of all variables.

City block distance. The Manhattan or *City-block* is the sum of the absolute differences between the pairs of corresponding values for two individuals (rather than the sum of the squares for Euclidean distance). It would be the distance needed to travel on a rectangular grid of streets and avenues between two locations, as Figure 1 shows, and as such is always larger than the Euclidean distance,

If the Manhattan distance measure is applied to binary variables and then divided by the number of variables, the resulting distance is identical to that obtained by applying the simple matching coefficient.

Mahalanobis distance. A generalisation of the Euclidean distance is the Mahalanobis distance. Essentially, the aim is to make appropriate allowance for differing variances of different variables and the correlations between pairs of those variables. The effect is that less weight is given to differences in values on variables with large variances (so the large differences on these do not overwhelm the overall measure) and additionally to compensate for "duplication of information" when two variables are highly correlated.

Other measures. As well as the two most commonly-used measures described above, many other possibilities have been proposed. Some of these are highly specific to particular applications and others are based on some theoretical statistical property (such as chis-quared distances, distances based on a measure of correlation, and special formulae when the variables are angles). The correlation-based measures are particularly useful when the numerical variable has been derived from an ordinal categorical one, since their use lessens the dependence upon the particular numerical scores used for the ordered categories. A full discussion is given by Gower and Legendre (1986).

Mixed variables. When some of the variables are numerical and some are categorical the possibilities are: (i) to convert some of the variables to be of the same type as the remainder and use one of the measures defined above or, (ii) to use a weighted sum of similarities calculated separately for variables of the same type or, (iii) to use a weighted sum of individual similarities calculated from pairs of variables in the most appropriate way. A special technique for the last of these, the Gower universal similarity coefficient, defines particular ways of doing this so that the resulting set of similarities has desirable mathematical properties, which would, however, be beyond the scope of this chapter.

To illustrate some of these measures we continue the example introduced above with measures of three numerical variables: *number of letters in the alphabet*, *number of speakers in millions* and *number of countries where the language is among the official languages.* The values are given in Table 10. Notice that these variables are measured in totally different units and we use this to illustrate the pitfalls present unless some form of standardisation is used.

The Euclidean distance between English and German as reflected by these variables is $\sqrt{[(26-30)^2+(360-120)^2+(7-3)^2]} = 240.067$. That between English and Polish is $\sqrt{[(26-30)^2+(360-40)^2+(7-1)^2]} = 320.081$. The complete distance matrix is given in Table 11.

For illustration and again temporarily ignoring the fact that the variables are measured in different units, the City-Block or Manhattan distance between English and German is $|26-30| + |360-120| + |7-3| = 4 + 240 + 4 = 248$. Here the vertical

Table 10. An example of three numerical variables measured on six languages

Language	Number of letters in the alphabet	Number of speakers in millions	Official language in number of countries
English	26	360	7
German	30	120	3
Dutch	26	28	6
Russian	33	155	5
Polish	32	40	1
Serbian	30	8.7	3

Table 11. Distance matrix of Euclidean distances between languages

	English	German	Dutch	Russian	Polish	Serbian
English	0	240.1	332.0	205.1	320.1	351.3
German	240.1	0	92.1	35.2	80.0	111.3
Dutch	332.0	92.1	0	127.2	14.3	19.9
Russian	205.1	35.2	127.2	0	115.1	146.3
Polish	320.1	80.0	14.3	115.1	0	31.4
Serbian	351.3	111.3	19.9	146.3	31.4	0

Table 12. Distance matrix of Manhattan distances between languages

	English	German	Dutch	Russian	Polish	Serbian
English	0	248.0	333.0	214.0	332.0	359.3
German	248.0	0	99.0	40.0	84.0	111.3
Dutch	333.0	99.0	0	135.0	23.0	26.3
Russian	214.0	40.0	135.0	0	120.0	151.3
Polish	332.0	84.0	23.0	120.0	0	35.3
Serbian	359.3	111.3	26.3	151.3	35.3	0

bars indicate the absolute value of the difference (i.e. larger minus smaller). Table 12 contains the complete distance matrix.

The Manhattan distances are generally a little greater than the Euclidean distances but a serious drawback to both is that it matters crucially what units are used for the individual variables. If the number of speakers is measured in billions rather than millions then the measures are not only changed substantially but the relative orderings of them are changed. So, a language which appeared to be "closest" to a given one (i.e. its nearest neighbour) might appear to be a distant neighbour if the units of one variable are changed. To demonstrate this, consider the data in Table 13.

Table 13. An example of three numerical variables measured on six languages

Language	Number of letters in the alphabet	Number of speakers in billions	Official language in number of countries
English	26	0.360	7
German	30	0.120	3
Dutch	26	0.028	6
Russian	33	0.155	5
Polish	32	0.040	1
Serbian	30	0.0087	3

These are the same as in Table 7 but with the number of speakers expressed in billions not millions. The table of Euclidean distances calculated from these figures is given in Table 14.

Table 14. Distance matrix of Euclidean distances between languages

	English	**German**	**Dutch**	**Russian**	**Polish**	**Serbian**
English	0	5.662	1.054	7.283	8.491	5.668
German	5.662	0	5.001	3.606	2.830	0.111
Dutch	1.054	5.001	0	8.127	11.012	7.019
Russian	7.283	3.606	8.127	0	5.15	5.146
Polish	8.491	2.830	11.012	5.115	0	4.031
Serbian	5.668	0.111	7.019	5.146	4.031	0

It is clear that not only the distances are changed in value but the relative ordering between the distances is substantially changed as well. For example, when measuring the number of speakers in millions (Table 11) the most similar pair is Russian and German with a Euclidean distance between them of 40.0. If we measure the number of speakers in billions, then the Russian-German pair is only the fourth closest; the closest pair is now German and Serbian. Similar changes are seen with the Manhattan distances but the Mahalanobis distances remain entirely unchanged and are identical to those in Table 13. We do not give these two tables here but the R code to produce them is provided in the Appendix. Similarly, it is easy to check that the Gower measure (given in Table 16) is unaltered by a change of units of a single variable.

This comparison illustrates why the basic similarity measures should only be used when all the variables are measured in the same units. However, situations arise frequently in practice where inevitably the variables are in different units. In this case, variables should be standardised. One method is to divide each variable by its standard deviation. This can be done in R with the function `scale(.)` setting `datamatrix<-scale(datamatrix)`, where `datamatrix` consists entirely of numerical values of the variables. A slightly better way is to use a Mahalanobis distance which additionally compensates for correlations between variables. It is not realistic to calculate a Mahalanobis distance by hand, even for the small example used here, but it can be done in a good computer package such as R. It does, however, require a little ingenuity since none of the standard libraries provides a function for calculating all pairwise distances between points. The complete table (calculated using the R code given in Appendix) is given in Table 15.

Finally, for mixed variables, i.e. when some are binary and some are numerical, the Gower coefficient essentially combines a measure based on binary variables with one based on numerical ones. There is a facility for specifying separate weightings for each variable but this goes beyond the scope of the description here. It can be added that the continuous variables are individually scaled but no allowance is made for

Table 15. Distance matrix of Mahalanobis distances between languages

	English	German	Dutch	Russian	Polish	Serbian
English	0	2.034	2.934	3.096	2.888	2.731
German	2.034	0	2.426	2.534	0.904	1.111
Dutch	2.934	2.426	0	3.091	2.992	1.684
Russian	3.096	2.534	3.091	0	2.889	2.434
Polish	2.888	0.904	2.992	2.889	0	1.376
Serbian	2.731	1.111	1.684	2.434	1.376	0

Table 16. Distance matrix of Gower distances between languages

	English	German	Dutch	Russian	Polish	Serbian
English	0	0.384	0.222	0.783	0.754	0.848
German	0.384	0	0.267	0.572	0.369	0.463
Dutch	0.222	0.267	0	0.706	0.545	0.625
Russian	0.783	0.572	0.706	0	0.427	0.236
Polish	0.754	0.360	0.545	0.427	0	0.342
Serbian	0.848	0.463	0.625	0.236	0.342	0

correlations. For completeness we give the resulting distance table (Table 16) which was calculated from R using the code given in the Appendix. The data used are those obtained by combining the categorical variables from Table 5 with the numerical ones in Table 10.

2.2.3 *Choice of similarity measure*

In many cases, the choice of similarity measure is also restricted by the type of data involved, i.e. categorical or numerical (although in some cases it is possible to convert one type to another). Where there is a choice between available measures, there may be little practical difference in the resulting analyses. The aspect that is most likely to affect the analysis, however, concerns the choice of weights given to the various properties measured. At its extreme, basing an analysis on similarity of meanings will be entirely different from an analysis based on numbers of syllables, i.e. whether the weights given to the properties 'meaning' and 'number of syllables' are 1 and 0, or 0 and 1, respectively. If it is desired to include both properties then it is up to the investigator to make a decision on the appropriate weights to use. Necessarily, this is a matter of subjective judgement determined by the objective of the analysis. Recall that giving equal weights is likewise a purely subjective judgement and needs to be justified by the study objectives. Particular care must be given to cases where variables are measured in different units. If all of the variables are numerical then they should be standardised to a similar scale. This is generally best done by using Mahalanobis distances (which also compensates for correlations) but a partial alternative is to use the Gower coefficient, which does standardise the variables. Even after standardisation it is still necessary to consider the relative weights of the variables and individual weights for each variable can be specified when using the Gower metric in function `daisy(.)`.

2.3 Cluster formation

All forms of cluster analysis are essentially algorithmic. That is, they are defined by the algorithm which actually produces "clusters" of similar objects. The choices that have to be made are, firstly, how to calculate the distance between clusters. Secondly, it has to be decided how to form clusters, i.e. whether to grow the clusters from the separate objects (agglomerative clustering), to divide the complete set of objects into successively smaller groups (divisive clustering), or to aim for a specific number of clusters and try to find the best division of the objects into that number of groups (*k*-means or partitioning clustering).[4] We will discuss each of these options in turn, starting with agglomerative algorithms.

2.3.1 *Agglomerative algorithms*

An *agglomerative* algorithm starts by regarding the *n* separate objects as *n* separate "clusters". Then, the first step is to join the two objects which are closest together to form a cluster of size 2. Next, either another pair is joined together to form another size 2 cluster, or if there is a single object closer to the first size 2 cluster than any other pair, then it is joined to form a size 3 cluster. The process carries on until finally all the objects are joined together. Different from the results of PCA or MDS, the outcome of a cluster analysis can be illustrated in one picture, a tree diagram (or dendrogram).

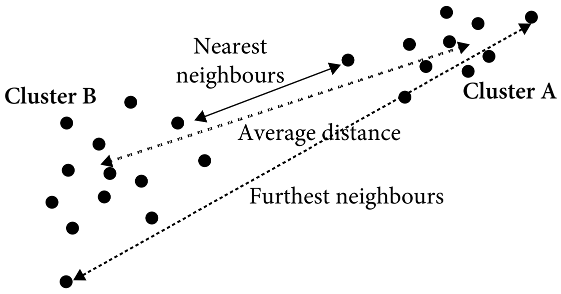

Figure 2. Different methods for defining distance between clusters

4. We will not discuss so-called Model-based approaches that assume a variety of data models and apply maximum likelihood estimation and Bayes criteria to identify the most likely model and number of clusters, e.g. the `Mclust()` function in the `mclust` package.

To measure the distance between objects or clusters, possibilities are the minimum, maximum and average distance between two points in different clusters. This is illustrated in Figure 2. "Average" could be the straightforward arithmetic mean, or the median, or some other measure based on the squared distance between cluster means divided by a measure of internal spread within the clusters, usually the average squared distance of elements from their cluster means. Using the squared distance rather than just the raw distance ensures that clusters are kept compact and do not become too spread out.

Let us explore these options in some detail and look at what family relations they propose for our cardinal numerals, the (dis)similarity between which is assessed on the basis of current spelling in terms of initial letters.

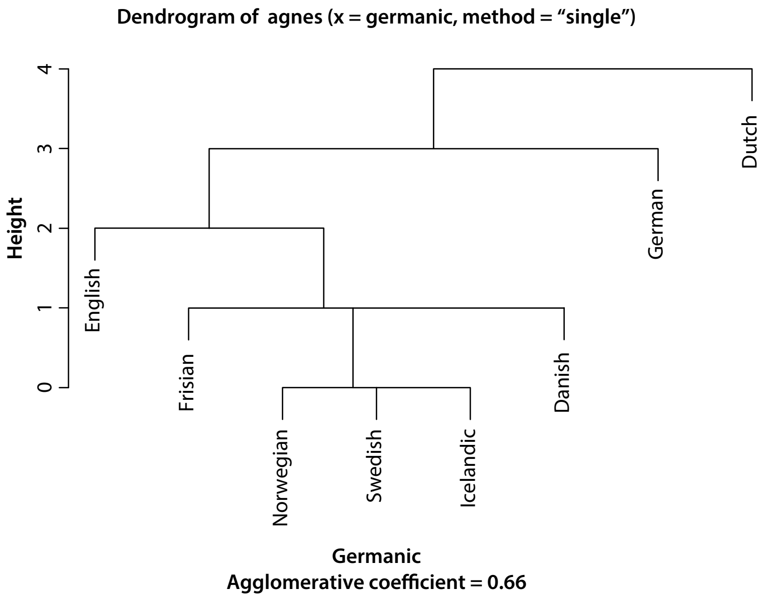

Figure 3. `Agnes` clustering of Germanic cardinal numerals using a Single linkage amalgamation algorithm[5]

A *nearest neighbour* or *single linkage* algorithm will tend to produce clusters where some objects are separated by a large distance although there is a string of objects in between them. This is the case because in single linkage, the distance between two

5. `agnes()` will let you start from a datamatrix of values of variables and then calculate a distance matrix internally, hence requiring specification of the distance metric, but also allows you to feed in a distance matrix. Other routines such as `hclust()` require you to calculate the distance matrix separately first using `dist()` or `daisy()`.

clusters is computed as the distance between the two closest elements in the two clusters. The result is rather like a railway network where two stations may be connected even though they are far apart, just because there are several intermediate stops. To be more specific, in the case of single linkage or nearest neighbour algorithms, the result reminds of a family tree with the most closely related (e.g. siblings) joined together at the first level, then separate families of siblings joined at the next level to form cousins and so on to produce a complete genealogy showing descendants from a single source. At each generation, the numbers of clusters (or families) become smaller but individual clusters contain members more distantly related. Furthermore, the clusters formed at one generation are subclusters of the next higher generation and so the analysis is a *hierarchical cluster analysis*. Let us look at the Germanic cardinal numbers from a single linkage point of view (Figure 3).

A dendrogram is a visual representation of the distances at which clusters are combined. It is read from top to bottom. Horizontal lines show joined clusters. The position of the line on the scale indicates the distance at which clusters are joined. When you read a dendrogram, you want to determine where the distances between clusters that are combined are large; hence you look for large distances between sequential vertical lines. From this dendrogram we can read that the most closely related are three of the Scandinavian languages: Norwegian, Swedish and Icelandic, in which cardinal numbers are spelled by and large identically. Next, Frisian and Danish are added. These appear on different sides of the first cluster but this is irrelevant for the dendrogram structure. The horizontal ordering of clusters in the dendrogram is arbitrary: a dendrogram can be compared to a children's mobile, suspended from a high point with clusters allowed to rotate freely. When it is taken down and laid on the table the horizontal ordering that the clusters end up in is accidental. In a third step, English is linked to the cluster including Frisian and Danish. Then German is added to the amalgam, and finally, Dutch, making Dutch the most dissimilar to the others of all Germanic languages included in the cardinal numbers sample.

Furthest neighbour or *complete linkage* algorithms use the maximum pairwise distance between elements as a measure of distance between clusters. These algorithms should generally produce more compact clusters, as can be seen in Figure 4 where English and Frisian as well as Dutch and German now each form a cluster.

Finally, algorithms based on some form of *average linkage* use the average distance between two points in different clusters; usually the average is taken as the straightforward arithmetic mean. These results will be intermediate in scale – in our case virtually identical to the single linkage result – between single and furthest linkage methods, as Figure 5 illustrates. Here, the only difference with the dendrogram resulting from single linkage (Figure 3) is to be found in the way in which Frisian and Danish are linked to the first cluster: while single linkage adds Danish first, followed by Frisian, average linkage adds both at the same time.

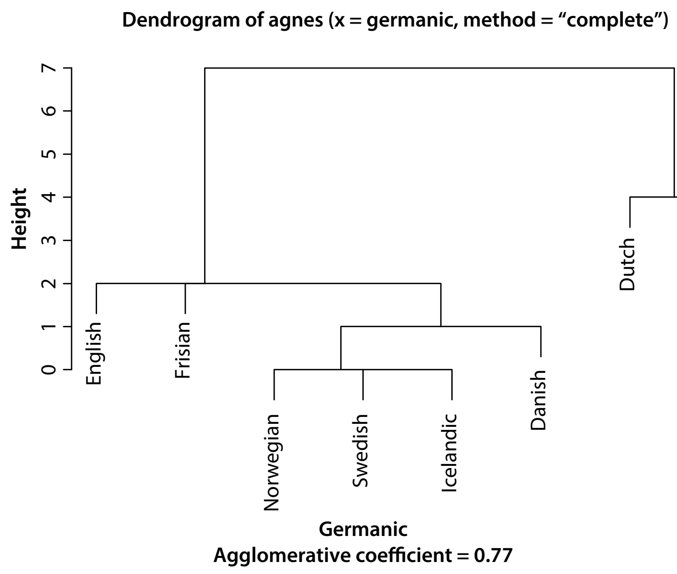

Figure 4. `Agnes` clustering of Germanic cardinal numerals using a Complete linkage amalgamation algorithm

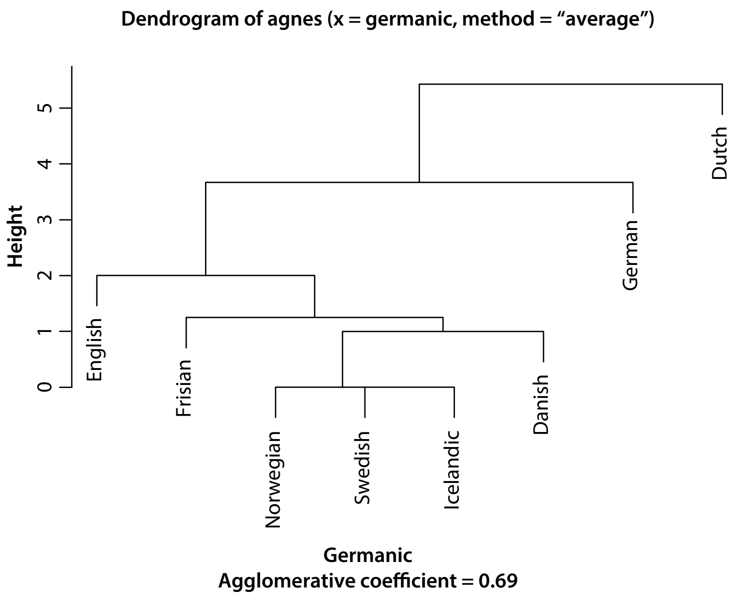

Figure 5. `Agnes` clustering of Germanic cardinal numerals using an Average linkage amalgamation algorithm

Generally, clusters produced by single linkage will be more "spread out" than those produced by complete linkage, with those from average linkage somewhere in between. This is because for a new member to join a cluster on single linkage it only has to be close to one single member of the cluster; this can result in a long string of elements, each close to its neighbours but one end a long way from the other. With complete linkage a new member has to be close to *all* other members of the group, keeping the clusters more compact.

The attentive reader will have noticed that all dendrogram plots list the agglomerative coefficient (AC), a measure of the clustering structure of the dataset that can range from 0 to 1. An AC close to 1 indicates that a very clear structuring has been found whereas an AC close to 0 indicates that the algorithm has not found a natural structure. This measure is sensitive to sample size, i.e. the value grows with the number of observations. For this reason, the AC should not be used to compare datasets of very different sizes. Given that we used the same datasets for all three clusterings, we can conclude that the combination of the Euclidean distance measure with the Complete Linkage algorithm yields the best results, the AC being equal to 0.77, thus outperforming Average Linkage (with AC = .69) and Single Linkage (with AC = .66).

One more method, *Ward's Method,* needs to be mentioned here. Ward's is a variant of the agglomerative methods outlined above and is widely used. This technique allows two clusters to merge if the increase in sum of squared distances of the members of the new cluster from their mean is smaller than for any other possible merger between two clusters. Use of squared distances penalises spread out clusters and so results in compact clusters without being as restrictive as complete linkage.

However, in many practical examples the differences between the resulting analyses applied to the same data are surprisingly superficial: with regards to the general picture, they are much the same with only a few differences of detail. In cases where there are substantial differences in the composition of clusters, thought must be given to why such differences occur and which of the methods is the most appropriate for the research questions of interest.

2.3.2 *Divisive algorithms*

In principle, the process of forming clusters can work in reverse, i.e. starting by dividing a single large family into two and so on, again leading to a hierarchical analysis. The aim at each step is to divide the most heterogeneous cluster into two more homogeneous clusters. Heterogeneity and homogeneity are measures based on the various alternatives suggested above. In most cases this requires exhaustive examination of all possibilities and so is computationally very expensive for even medium-sized data sets. An exception is when the variables are all binary, in which case splits can be made using just one variable at a time. This *monothetic divisive method* takes individual variables in sequence rather than amalgamating them into an average and divides a cluster so that all elements in the two subclusters have a common value of a particular variable, i.e. if the split is made on the alphabet variable then all elements in one subcluster will have value 1 for Latin script and in the other subcluster they will have value 0 of that variable, or Cyrillic script. Choice of which variable to choose has to be made on some optimality criterion, the discussion of which is beyond the scope of this chapter.

2.3.3 K-means *clustering*

An alternative divisive (but non-hierarchical) approach is to specify the desired number of clusters, k say, and search for a division of the n objects into k groups which is "best" in some defined sense. Such methods are usually referred to as *k-means* procedures and are within the class of *optimization methods*. A common measure of "best" is to require the average distance between objects in a cluster and their cluster centroid to be minimised in comparison to the average distance of cluster centroids from the centroid of all objects. Here "centroid" usually indicates the mean value of the cluster, but could be some more general measure of the "center" of the cluster. The measure used may be based on squared Euclidean distances rather than actual Euclidean distances to penalise allocations where one or two objects are far away from the centroid. This ensures more compact clusters which is often a goal of the analysis.

In theory one might try to examine all possible arrangements of the n objects into k clusters and see which is "best" but the number of possible allocations explodes rapidly with increasing n and k. For ten objects and three clusters, the number of allocations is a little less than ten thousand. Consequently, all available computer techniques rely on iterative procedures. For example, the first step is to divide the objects arbitrarily into k groups, calculate the group centroids and then reallocate an object to a different group if it is closer to that group's centroid than to its own, re-calculating group centroids after each re-allocation. Generally, such a technique will eventually converge (meaning that no more re-allocations are required) and the final result will be optimal or close to optimal. There is no mathematical reason for the procedure to converge on the absolute optimal solution, so most computer packages allow the procedure to be tried several times with different randomly selected allocations.

This description begs the question of how to choose k, the desired number of groups. In the absence of some value being determined *a priori* by the objectives of the research, essentially the answer is to try several values and see which produces the most "sensible answers". One approach might be to try a more rapid technique such as single link agglomerative clustering to suggest a suitable number of clusters and then improve on the allocation.

2.3.4 *Differently named methods*

There are several named techniques which appear to have special meanings or procedures, but are in reality merely particular computer implementations of the procedures described above. Many of these are contained in the widely used R library `cluster` and are actually functions that have been imaginatively given female first names (see Kaufman and Rousseeuw 1990 for details).

The basic function for agglomerative clustering in the library cluster is `agnes(.)` [from AGglomerative NESting]. This function can take (optional) arguments to determine the particular type of algorithm to be used (i.e. method = "single", "average", "complete" or "ward"). Within this package there are also functions available for other single non-agglomerative techniques, (i) `diana(.)` [from DIvisive ANAlysis Clustering] performs divisive hierarchical clustering, (ii) `mona(.)` [from MONothetic Analysis clustering of binary variables] is the form of divisive clustering appropriate when the values of all the variables are 0 or 1, while (iii) `pam(.)` [from Partitioning Around Medoids] and (iv) `clara(.)` [Clustering Large Applications] are both particular forms of *K*-means clustering; the latter is useful when the number of objects is very large (more than several thousand) while in other cases `pam(.)` is preferred. Finally, (v) `fanny(.)` [from Fuzzy ANalYsis clustering] implements a rarely used method which allows objects to belong partially to each of several different clusters. In fuzzy clustering, each point has a degree of belonging to clusters rather than belonging completely to just one cluster. Thus, points on the edge of a cluster may be in the cluster to a lesser degree than points in the centre of cluster.

Now, back to our example. What happens when we make the task a bit more difficult? We can add in more languages from one family, for example, languages that we know relate in a non-straightforward way to the core languages. As Figure 6 shows, cluster analysis can deal with such cases, keeping the newly added Celtic languages

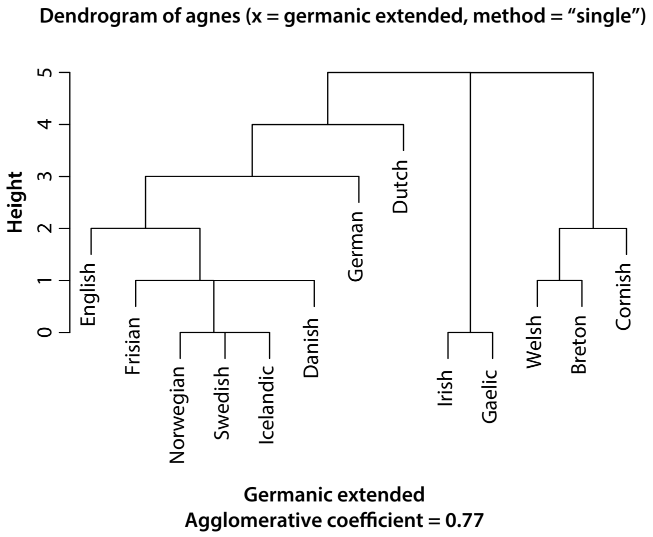

Figure 6. `Agnes` clustering of Germanic (extended) cardinal numerals using a Single Linkage amalgamation algorithm

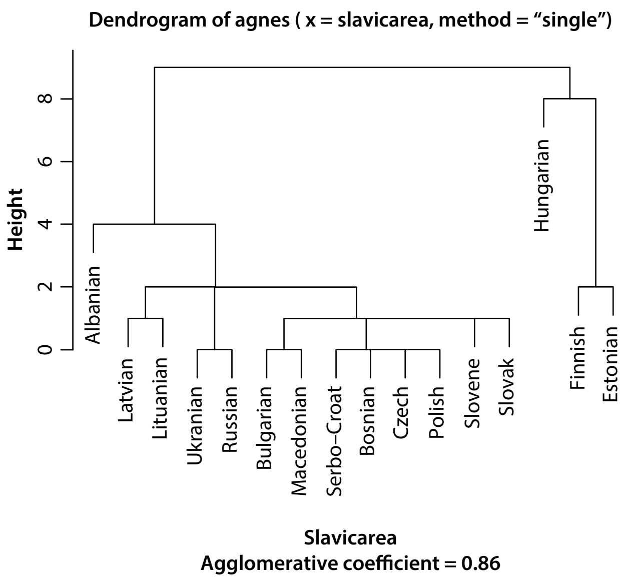

Figure 7. `Agnes` clustering of Slavic area cardinal numerals using a Single linkage amalgamation algorithm

distinct from the Germanic core that remains unchanged, while discovering similarities between the Celtic languages.

In other cases, we might not want to limit our sample to a group of languages that are known to relate to each other but, for example, to all languages spoken in a particular area. Using areal data rather than family data shows how clustering deals with expected outliers. As an example, we include Baltic, Finnic and Ugric languages in the Slavic mix and obtain the dendrogram presented in Figure 7.

Clearly, Latvian and Lithuanian share most similarities with the Slavic languages, as does, to some extent, Albanian. The further we move to the borders of the Slavic lands, towards Hungary in the South East and Estonian and Finnish in the North, the starker the differences become.

We can also add in languages from two different families and ask whether different families can reliably be retrieved? This is done in Figure 8, which clearly shows a division between Germanic and Slavic families.

The more information we include in the cluster analysis, the more complex the outputted dendrograms become. At the same time, as the number of clusters increases, the more urgently the question poses itself of how to decide on the optimal numbers of clusters.

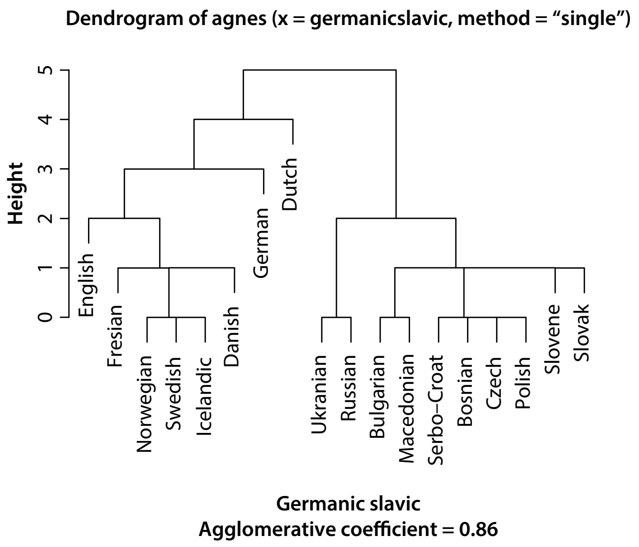

Figure 8. `Agnes` clustering of Germanic and Slavic cardinal numerals using a Single linkage amalgamation algorithm

2.4 Deciding on the "optimal" number of clusters

Cluster analysis presents the problem of how many factors, dimensions, or clusters to keep. Although "look for cluster groupings that agree with existing or expected structures" and "pick the one solution you like best" is not a frivolous comment in the context of cluster analysis, the utility of clusters must be assessed regardless of the method used to form the clusters. One rule of thumb for this is, as we mentioned before, to look at the height bar and to choose a place where the cluster structure remains stable for a long distance, i.e. no new clusters are added, such as, for example, between the Germanic and the Slavic group in Figure 8. Below we list three further criteria:

1. *Size*. All clusters should have enough cases to be meaningful. In the case of *K*-means clustering, where the researcher sets the number of clusters to be formed, one or more very small clusters indicates that the researcher has requested too many clusters. Analysis resulting in a very large, dominant cluster may indicate too few clusters have been requested.
2. *Meaningfulness*. Ideally, the meaning of each cluster should be readily intuited from the constituent variables used to create the clusters. Variable importance plots are one method of making this assessment.
3. *Criterion validity*. The cross-tabulation of the cluster identification names or numbers by variables that are known (from theory or prior research) to correlate with the concept which clustering is supposed to reflect should reveal the expected level of association.

If these criteria are not met, care should be taken when interpreting the results of the cluster analysis. It may be the case that an inappropriate distance measure has been selected or that the hypothesized conceptual basis for clustering does not exist, resulting in arbitrary clusters.

One way to assess the confidence one may have on the obtained cluster solution more objectively includes replicating the analysis on a different dataset or on subsets of the data to see if the structures emerge consistently. This is one of two relatively straightforward ways of validating a cluster analysis that are explained in the following section.

2.5 Calculating the "fit" of/validating a cluster solution

Let us consider another example, this one based on the pronunciation of Slavic numerals, listed in Table 3. The similarity matrix between languages was calculated by counting the number of matches of first letters as pronounced. This was converted to a distance matrix by subtracting each similarity from 10, as described in Section 2.2.

The (resulting) dataset was subjected to a cluster analysis with Ward's clustering algorithm, using the `hclust` package.[6] The output dendrogram is presented in Figure 9.

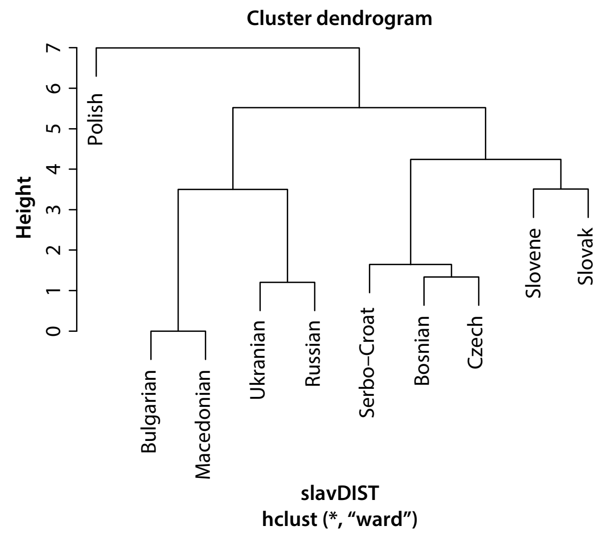

Figure 9. `hclust` clustering of Slavic cardinal numerals as pronounced using Ward's amalgamation algorithm

6. We switch to `hclust` here to maintain consistency with Section 2.5 where we illustrate `pvclust` that relies internally on `hclust`.

In the next two sections we will examine the robustness of our clustering result in two ways, i.e. by means of examining silhouette widths (Section 2.5.1), as well as by running analyses on random subsamples of the dataset (Section 2.5.2).[7]

2.5.1 *The Silhouette validation technique (Rousseeuw 1987)*

Silhouettes are a graphical aid to the interpretation and validation of cluster analysis. Each cluster is represented by a so-called *silhouette*, which is based on the comparison of its tightness and separation. This silhouette shows which objects lie well within their cluster, and which ones are merely somewhere in between clusters. The silhouette value measures the degree of confidence in the clustering assignment of a particular observation, with well-clustered observations having values near 1 and poorly clustered observations having values near −1. If the silhouette value is close to 1, the sample is "well-clustered", i.e. it was assigned to a very appropriate cluster. If the silhouette value is around zero, that sample could be assigned to another cluster as well, and the sample lies equally far away from either option. If the silhouette value is close to −1, the sample is "misclassified" and is somewhere in between the clusters.

The entire clustering is displayed by combining the silhouettes into a single plot, allowing an appreciation of the relative quality of the clusters and an overview of the data configuration. The overall average silhouette width for the entire plot is the average of the *S(i)* for all objects in the whole dataset. The largest overall average silhouette indicates the best clustering. Since the average silhouette width provides an evaluation of clustering validity, it might be used to select an "appropriate" number of clusters.

For our dataset, we see that we can be reasonably sure that Ukrainian, Russian, Bulgarian and Macedonian form a coherent group, with the silhouette width for that cluster of four languages being 0.43 (Figure 10). We are less sure that Serbo-Croat, Bosnian, Czech, Slovak and Slovene, if combined into one cluster, indeed belong together, their silhouette width being 0.20. Polish, finally, is kept separate, and the silhouette value for that separate cluster is 0, indicating that Polish could well have been assigned to a different cluster. The overall average silhouette width remains at a low 0.27, indicating that the proposed clustering as a whole may not be too sensible; this is very likely due to the way in which the pronunciation data was coded. Maybe a

7. The function `cluster.stats()` in the `fpc` package (Hennig 2010) provides a mechanism for comparing the *similarity* of two cluster solutions using a variety of validation criteria, including the average silhouette widths treated below. To be precise, `cluster.stats()` computes a number of distance based statistics which can be used for cluster validation, comparison between clusterings and decision about the number of clusters: cluster sizes, cluster diameters, average distances within and between clusters, cluster separation, average silhouette widths, the Calinski and Harabasz index, the best distance based statistics to decide about the number of clusters in a study of Milligan and Cooper (1985), Hubert's gamma coefficient, the Dunn index and two indexes to assess the similarity of two clusterings, namely the corrected Rand index and Meila's VI. Another package of interest in this respect is `clValid` (Brosk et al. 2011).

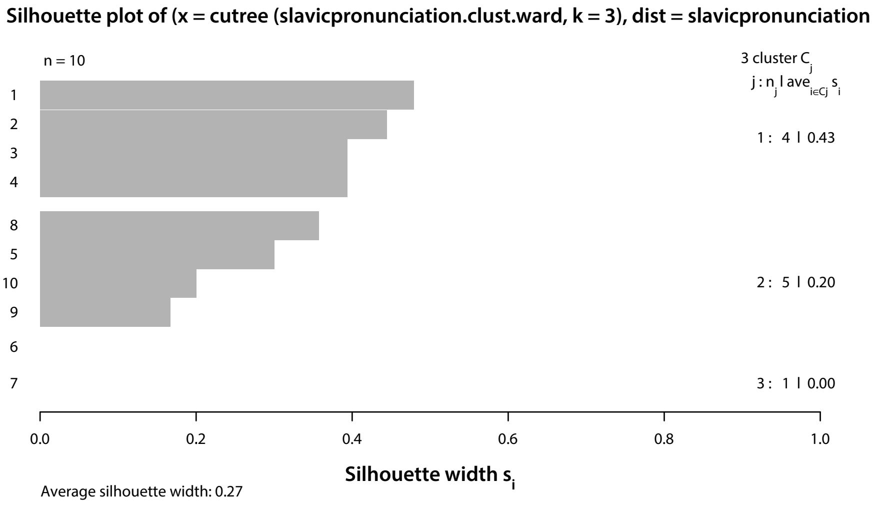

Figure 10. Silhouette plot for a three-cluster solution of Slavic cardinal numerals as pronounced

2-cluster solution would fit the data better? In the next section we will introduce a technique that can assess this (un)certainty in terms of the more familiar *p*-values.

2.5.2 `pvclust`

The uncertainty in hierarchical cluster analyses can be assessed, even if no independent test-sample is available, by means of *p*-values or values between 0 and 1 that indicate how strongly the clusters are supported by the data. `pvclust` is an R package that performs hierarchical cluster analysis on a numeric data matrix or data frame via the function `hclust` and automatically computes *p*-values for all clusters contained in the clustering of the original data.[8] It also provides graphical tools such as the plot

8. The function `pvclust`, used for assessing uncertainty in clustering and illustrated below, is written so that it will only accept data entered as values of numerical variables (unlike `agnes` used so far which offers the choice of entering a distance matrix directly). Further, a method of calculating the distance matrix from the values of the numerical variables must be selected from a limited set of choices of Euclidean, Manhattan or binary. For the purposes of being able to illustrate `pvclust` we have fudged raw data by first calculating approximate Euclidean coordinates from the distance matrix using the R function `cmdscale` and then use these to calculate an approximate distance matrix. The result corresponds closely with that obtained based on actual numbers of matches and non-matches, no entry differing by more than 1.5 on a scale of distances between 0 and 10. This is not a technique we would usually recommend and we present it here only so that we can illustrate the use of `pvclust` for assessing uncertainty in clustering.

function or the `pvrect` function which highlights clusters with relatively high/low *p*-values.

For each cluster in hierarchical clustering, *p*-values are calculated via multiscale bootstrap resampling, a computer-based way of simulating similar datasets. `pvclust` provides two types of *p*-values: the AU (Approximately Unbiased) *p*-value (on the left, normally in red) and BP (Bootstrap Probability) value (on the right, normally in green). The AU *p*-value, which is computed by multiscale bootstrap resampling, is a better approximation to unbiased *p*-value than the BP value computed by normal bootstrap resampling. Going into details about bootstrap resampling methods is beyond the scope of this chapter.

Clusters that are highly supported by the data will have large *p*-values. `pvclust`, by and large, confirms the assessment based on silhouette widths:[9] we should not be too sure that the larger clusters exist outside of this dataset. When using a different dataset that likewise assesses the languages on the basis of the pronunciation of Slavic numerals between 1 and 10, we can be 100% sure, however, that Bulgarian and Macedonian would be grouped together and 97% certain that Bulgarian, Macedonian, Ukrainian and Russian would be clustered together, while being 95% sure that Bosnian, Serbo-Croat and Czech would end up in the same group.

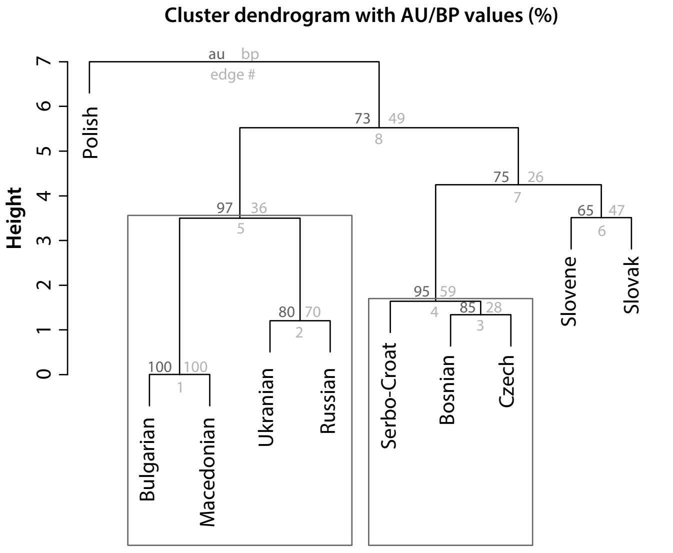

Figure 11. `pvclust`, *p*-values for a dendrogram of Slavic cardinal numerals as pronounced. Distance: Euclidean; method: Ward. 1000 bootstraps

9. Be aware that `pvclust` clusters columns, not rows. This necessitates transposing the data before running the procedure. The code is provided in the Appendix to this chapter.

3. By way of conclusion

This brief account could do little more than introduce Cluster Analysis as an intuitive tool for describing and categorizing linguistic data in a systematic and reproducible way that displays the outcome of the analysis visually in one dendrogram. We have emphasized that this does not give a totally "objective" method. On the contrary, there are many steps where subjective choices have to be made. We have highlighted these and discussed the impact that these may have on the ensuing results. The attentive reader will also have noticed that, different from its direct competitors PCA and MDS, Cluster Analysis does not provide information on the relative importance of variables or underlying factors; it is up to the analyst to extract these from the data (see Divjak and Gries 2006 for the implementation of some suggestions by Backhaus *et al.* 1996: 310–312).

We did not feel it was appropriate here to go into the theoretical statistical background underpinning the methods. For these the reader is referred to Everitt *et al.* (2011) for the statistical side and to Johnson (2008), Baayen (2006) or Gries (2009) for an account in a linguistic setting. Neither have we done more than give the minimum of details of the R code and R language needed to implement them. For these the reader is referred to the textbooks and to the R system itself. Readers who invest some time trying the R system for themselves will quickly discover the power of its internal Help system. We have listed the names of the primary functions used for the various types clustering and these are sufficient to find details of how they are used and worked examples giving illustrations from the Help system. The reader is encouraged to work through these in detail and be assured that the initial effort will be quickly repaid.

References

Alviar, J. J. (2008). Recent advances in computational linguistics and their application to biblical studies. *New Testament Studies*, 54(1),139–159 DOI: 10.1017/S0028688508000088

Baayen, R. H. (2008). *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge: Cambridge University Press. DOI: 10.1017/CBO9780511801686

Backhaus, K., Erichson, B., Plinke, W., & Weiber, R. (1996). *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 8th edition. Berlin; Heidelberg; New York: Springer.

Brock, G., Pihur, V., Datta, S., & Datta, S. (2011). clValid: Validation of clustering results. *Journal of Statistical Software*, 25(4), March 2008. R package version 0.6-2. <http://CRAN.R-project.org/package=clValid>.

Divjak, D., & Gries, St. Th. (2006). Ways of trying in Russian: Clustering behavioral profiles. *Journal of Corpus Linguistics and Linguistic Theory*, 2(1), 23–60.

Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Cluster analysis*. 5th edition. Oxford: Wiley. DOI: 10.1002/9780470977811

Gower, J., & Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1), 5–48. DOI: 10.1007/BF01896809

Gries, St. Th. (2009). *Statistics for linguistics with R: A practical introduction*. Berlin: Mouton de Gruyter. DOI: 10.1515/9783110216042

Harnad, S. (2005). To cognize is to categorize: Cognition is categorization. In C. Lefebvre & H. Cohen (Eds.), *Handbook on categorization* (pp. 19–43). Oxford & London: Elsevier. DOI: 10.1016/B978-008044612-7/50056-1

Hennig, C. (2010). fpc: Flexible procedures for clustering. R package version 2.0-3. <http://CRAN.R-project.org/package=fpc>.

Johnson, K. (2008). *Quantitative methods in linguistics*. New York: Wiley-Blackwell.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis (Series in Applied Probability and Statistics)*. New York: Wiley-Blackwell.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50, 159–179.

R Development Core Team (2008). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. <http://www. R-project.org>.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1), 53–65. DOI: 10.1016/0377-0427(87)90125-7

Shaw, D. (1974). Statistical analysis of dialectal boundaries. *Computers and the Humanities*, 8, 173–177. DOI: 10.1007/BF02402137

Suzuki, R., & Shimodaira, H. An R package for hierarchical clustering with p-values. Retrieved from <http://www.is.titech.ac.jp/~shimo/prog/pvclust/t/> [Accessed 25 May 2012].

Tryon, R. C. (1939). *Cluster analysis*. New York: McGraw-Hill.

Wichern, D. W., & Johnson, R. A. (2007). *Applied multivariate statistical analysis*. Englewood Cliffs: Prentice-Hall.

Appendix

R code to create the distance matrices (tables) and dendrograms (figures) shown in this chapter

In this final section we provide the code necessary to carry out the operations illustrated in this chapter. Those with little familiarity with R can run the code by copying it, bit by bit, into the R console, although it may be helpful to consult a basic introduction for R first. The data and R sessions can be downloaded from http://dx.doi.org/10.1075/hcp.43.16div.additional.

We first provide the R code for calculating the tables of distances using the various measures we described above in Tables 5 through 14.

The easiest function to use is `dist(.)` which is in the `stats` library and so is automatically loaded for every R session. The `dist(.)` function allows computation of distances based on the simple matching and Jacard coefficients for binary data and Euclidean and Manhattan distances for continuous data. Which distance is calculated is determined by specifying it the argument `method`. It should be noted that if `method` is given as `binary` then distances are calculated as the complement of the Jacard coefficient. To obtain distances based on the complement of a simple matching coefficient (i.e. giving equal weight to positive and negative matches) then the Manhattan method should be chosen. This is not mentioned in the R documentation.

A more flexible function is `daisy(.)` which is in the `cluster` library and must be loaded separately. This allows all of the previously mentioned coefficients plus the Gower coefficient. The distance measures for binary variables are handled by specifying `gower` as the metric and then choosing symmetric or asymmetric for the simple or Jacard coefficients. Care must be taken when using the `daisy(.)` function that the classes of variables are declared correctly as factor or numerical. The R help system gives full details of this.

Neither of these functions allows calculation of the Mahalanobis distances and although there is a `mahalanobis` function in the `stats` library it is not designed to calculate the complete set of pairwise distances between points. We give some code to produce the complete pairwise distances and experienced R users may recognise that the key idea is to standardise the data matrix before calculating the Euclidean distances.

```
# load the library needed for accessing the function daisy(.)
library(cluster)
options(digits=3)

# the following code assumes that all data files are in the
# current R working directory; if that is not the case, replace
# file=“filename.txt” with choose.files() that will let
# you browse your computer to select the file of interest

# read in the data from Table 6 from a comma separated file
# with first line as header and first column as row names
excategoric<-read.csv(file=“excategoric.csv”,header=T,row.names=1)
# calculate distances
dist(excategoric,method=“manhattan”)/2
dist(excategoric,method=“binary”)

# read in data in Table 10 from a comma separated file with
# first line as header and first column as row names
exnumerical<-read.csv(file=“exnumerical.csv”, header=T, row.names=1)
# calculate Euclidean distances
dist(exnumerical,method=“euclidean”)
# calculate Manhattan distances
dist(exnumerical,method=“manhattan”)
# calculate Mahalanobis distances
v=var(exnumerical)
x<-svd(v)
vroot<-x$u%*%diag(sqrt(x$d))%*%t(x$v)
dist(as.matrix(exnumerical)%*%solve(vroot))
```

```
# read in complete data of mixed categorical and numerical
# variables from a comma separated file with first line as
# header and first column as row names
exmixed<-read.csv(file="exmixed.csv",header=T,row.names=1)
# calculate the Gower distances
daisy(exmixed,metric="gower")

# Repeat calculations of distances after converting data
# to billions of speakers, then repeat calculations above
exnumerical[,2]<-exnumerical[,2]/1000
# now have numbers of speakers in billions
# calculate Euclidean distances
dist(exnumerical)
# calculate Manhattan distances
dist(exnumerical,method="manhattan")
# calculate Mahalanobis distances
v=var(exnumerical)
x<-svd(v)
vroot<-x$u%*%diag(sqrt(x$d))%*%t(x$v)
dist(as.matrix(exnumerical)%*%solve(vroot),method="euclidean")
```

Next, we provide the R code for running the cluster analyses we described above in Figures 3 through 11.

```
# reads in .txt files, recognizes them as dissimilarity
# matrices and converts them to Rdata files
# this code assumes that all data files are in the current R
# working directory; if that is not the case, replace
# file="filename.txt" with choose.files() that will let
# you browse your computer to select the file of interest

x<-read.table(file="Germanic.txt",header=T,row.names=1)
# reads in .txt file
germanic<-as.dist(x)
# coerces the object x to be of class "distance",
# i.e. makes sure that the dissimilarity matrix is read
# in as a ready dissimilarity matrix
save(germanic, file="germanic.Rdata")
# saves it as Rdata file

x<-read.table(file="Germanic_extended.txt",header=T,row.names=1)
germanicextended<-as.dist(x)
save(germanicextended, file="germanicextended.Rdata")

x<-read.table(file="Slavic area.txt",header=T,row.names=1)
slavicarea<-as.dist(x)
save(slavicarea, file="slavicarea.Rdata")
```

```
x<-read.table(file="Germanic_Slavic.txt",header=T,row.names=1)
germanicslavic<-as.dist(x)
save(germanicslavic, file="germanicslavic.Rdata")

x<-read.table(file="Slavic pronunciation.txt",header=T,row.names=1)
# turns a similarity matrix into a distance matrix
slavicpronunciation<-10-as.dist(x)
save(slavicpronunciation, file="slavicpronunciation.Rdata")

# loads the libraries needed to run the cluster analyses
# runs the cluster analyses described and outputs
# the dendrograms

library(cluster)
library(amap)

#figure 3
load("germanic.Rdata") # loads the dataset
germanic.clust.single<-agnes(germanic,method="single")
# clusters the data using single linkage; no distance
# metric needs to be specified since the data are entered
# as dissimilarities
plot(germanic.clust.single) # outputs the dendrogram

#figure 4
load("germanic.Rdata")
germanic.clust.compl<-agnes(germanic,method="complete")
plot(germanic.clust.compl)

#figure 5
load("germanic.Rdata")
germanic.clust.average<-agnes(germanic,method="average")
plot(germanic.clust.average)

#figure 6
load("germanicextended.Rdata")
germanicextended.clust.single<-agnes(germanicextended,method="single")
plot(germanicextended.clust.single)

#figure 7
load("slavicarea.Rdata")
slavicarea.clust.single<-agnes(slavicarea,method="single")
plot(slavicarea.clust.single)
```

```
#figure 8
load("germanicslavic.Rdata")
germanicslavic.clust.single<-agnes(germanicslavic,method="single")
plot(germanicslavic.clust.single)

#figures 9, 10 & 11
# loads the library
library(pvclust)

# loads the dataset
load("slavicpronunciation.Rdata")

# runs a multidimensional scaling to obtain
# approximate Euclidean coordinates
slavMDS<-cmdscale(slavicpronunciation,k=9,eig=TRUE,x.ret=TRUE)
slavcoords<-slavMDS$points[,1:6]
slavDIST<-dist(slavcoords,method="euclidean")

# runs cluster analysis with hclust using
# Ward's amalgamation strategy
slavicpronunciation.clust.ward<-hclust(slavDIST, method="ward")
#plots the dendrogram
plot(slavicpronunciation.clust.ward)

# cuts the dendrogram into 3 clusters
sil<-silhouette(cutree(slavicpronunciation.clust.ward,k=3),
slavicpronunciation)
# plots the silhouette widths
plot(sil, nmax=80, cex.names = 0.5)

# clusters the data using ward's amalagamation algorithm
# while taking 1000 random samples
pron.slavic.pvclust<-pvclust(t(slavcoords),
method.dist="euclidean", method.hclust="ward", nboot=1000)
# plots the results
plot(pron.slavic.pvclust)
# draws rectangles around the highest cluster(s) that are
# distinguished at the .05 level
pvrect(pron.slavic.pvclust, alpha=0.95)
```

A simple way of creating dissimilarities from similarities for the numerals dataset is subtracting the difference value from the maximum similarity of 10, for example, as was done for the Slavic pronunciation dataset above

```
slavicpronunciation.dis<-10-slavicpronunciation
```

It may be necessary to transpose the rows and columns since `pvclust` clusters what is in the rows. This is done as follows:

```
pron.slavic<-t(pron.slavic)
```

If a dataset contains large numbers of items to be clustered, the interactive function `identify.hclust` can be used to list the members of a cluster by pointing with the mouse and clicking (with the right button) on the vertical bars (a.k.a. ancestor lines).