6 Historical Linguistics

Though there are many opportunities to use quantitative methods in historical linguistics (e.g. the Rigveda exercise at the end of chapter 5), this chapter will focus on how historical linguists take linguistic data and produce tree diagrams showing the historical "family" relationships among languages.

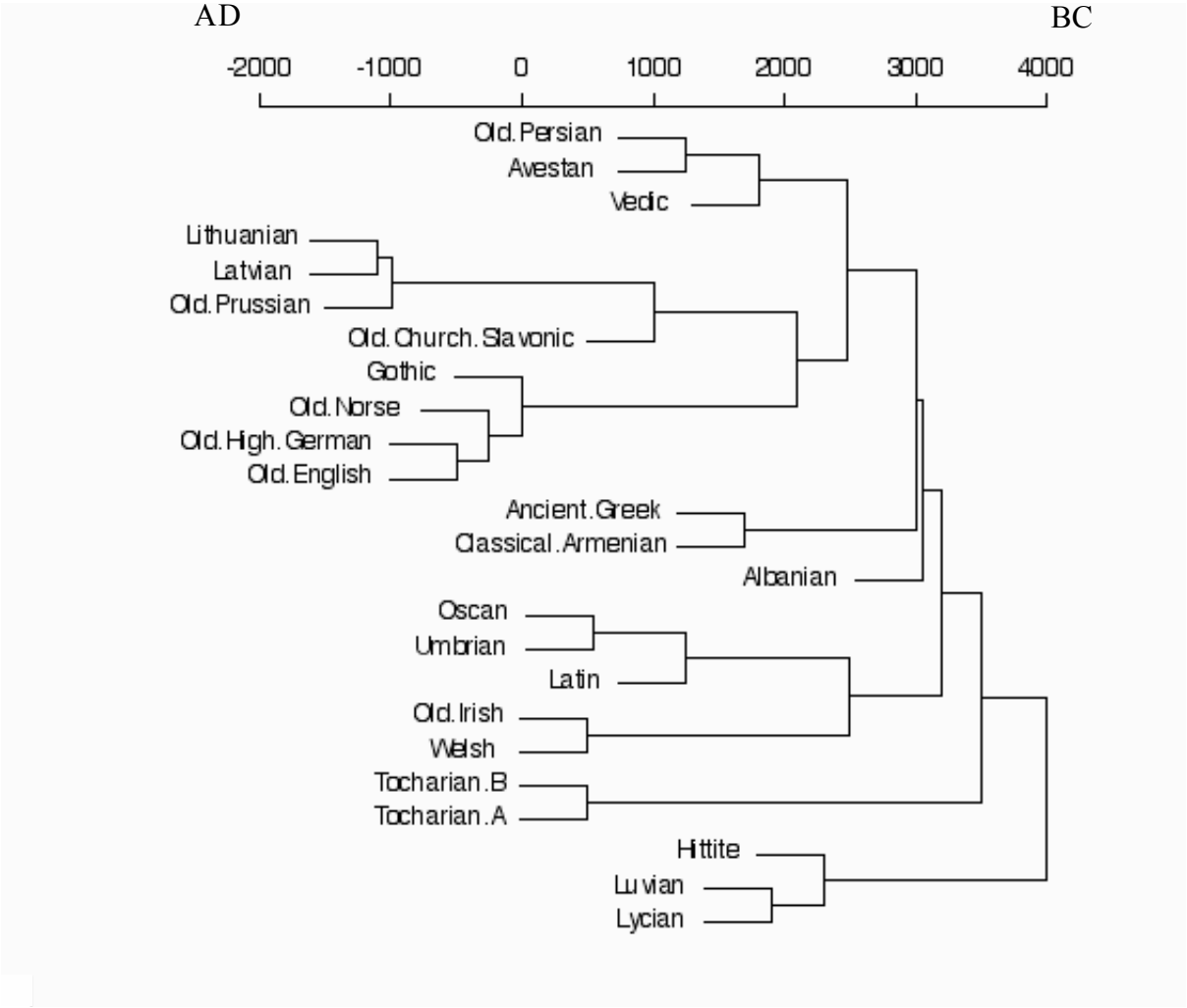

Figure 6.1. A phylogenetic tree of Indo-European proposed by Nakhleh, Ringe, & Warnow (2005).

For example, figure 6.1 shows a phylogenetic tree of Indo-European languages (after figure 6 of Nakhleh, Ringe & Warnow, 2005). This tree embodies many assertions about the history of these languages. For example, the lines connecting Old Persian and Avestan assert that these two

languages share a common ancestor language and that earlier proto-Persian/Avestan split in about 1200BC. As another example consider the top of the tree. Here the assertion is that proto Indo-European, the hypothesized common ancestor of all of these languages, split in 4000 BC in two languages - one that would eventually develop into Hittite, Luvian and Lycian, and another ancient language that was the progenitor of Latin, Sanskrit, Greek and all of the other Indo-European languages in this sample. The existence of historical proto languages is thus asserted by the presence of lines connecting languages or groups of languages to each other, and the relative height of the connections between branches shows how tightly the languages bind to each other and perhaps the historical depth of their connection (though placing dates on proto-languages is generally not possible with out external historical documentation).

6.1 Cladistics: Where linguistics and evolutionary biology meet

The aims of historical linguistics are to trace the history of languages and offer explanations of language change. Central to this research is the observation that languages can be grouped into families on the basis of their shared history - so that two distinct languages in the present (e.g. French and Italian) can be grouped together in a family tree because they are both decendants of Latin, and thus "genetically" related, arising from the same source.

This conception of language history assumes (1) that it is possible to identify groups of languages that are related by descent to a common ancestor language, (2) that languages change over time, and (3) that a language will split into daughter languages when the speakers of the language split into different geographically or culturally separated communities. Analogs of these three assumptions also underlie the construction of family trees in evolutionary biology. Consequently, historical linguists and evolutionary biologists have been exploring methods of quantitative analysis that they call "cladistics" (Gk. κλαδοσ - "branch").

From the point of view of general quantitative methods in data analysis, cladistics is interesting because in these studies biologists and linguists have, to some extent, made simultaneous independent discoveries, arriving at data analysis techniques that have also been explored in other areas of science (and are available in R). However, the unique set of constraints on biological and linguistic evolution has resulted in unique analysis strategies. Thus, we will see in this chapter that special purpose software is being used for state of the art cladistic analysis and although the functionality of this software could in principle be replicated in R, so far it hasn't been.

6.2. Clustering on the basis of shared vocabulary

One quantitative approach to judging historical relatedness was reported by Dyen, Kruskal, & Black (1992). They took the number of shared cognates in a list of 200 cross-linguistically common words (Swadesh, 1952) as a measure of overall similarity among 84 Indoeuropean

languages and then used a cluster analysis to produce a phylogenetic tree of Indo-European.

Table 6.1. Patterns of cognation among some of the languages in the Dyen et al. data set for five of the two hundred meanings listed in that set.

	"animal"		"bark (of a tree)"		"bird"		"black"		"to breathe"	
Dutch	dier	1	schors	3	vogel	4	zwart	1	ademen	2
Danish	dyr	1	bark	1	fugl	4	sort	1	aande	5
Swedish	djur	1	bark	1	fagel	4	svart	1	andas	5
German	tier	1	rinde	2	vogel	4	schwarz	1	atmen	2
English	animal	B	bark	1	bird	B	black	6	to breathe	B
Slovenian	zver	2	kora	3	ptica	5	crn	3	dihati	1
Russian	zver	2	kora	3	ptica	5	cernyj	3	dysat	1
Polish	zwierze	2	kora	3	ptak	5	czarny	3	oddychac	1
Czech	zvire	2	kura	3	ptak	5	cerny	3	dychati	1
Greek Mod	zoo	2	fludha	6	pouli	1	mauros	7	anapneo	3
Kashmiri	janawar	3	del	B	pankhi	3	kala	2	shah hyonu	4
Gujarati	jenawer	3	chal	4	penkhi	3	kalu	2	swaslewo	4
Marathi	jenaver	3	sal	4	peksi	3	kala	2	svas ghene	4
Hindi	janver	3	chal	4	peksi	3	kala	2	sas lena	4
Waziri	dzanawar	3	patikai	5	marghai	6	tor	5	saya	4
Portuguese	animal	4	cortica	3	ave	2	negro	4	respirar	6
Spanish	animal	4	corteza	3	ave	2	negro	4	respirar	6
Italian	animale	4	scorza	3	uccello	2	nero	4	respirare	6
French	animal	4	ecorce	3	oiseau	2	noir	4	respirer	6

For example, a portion of the Dyen et al. data set is shown in table 6.1. There are four cognate sets for words meaning “animal” among the 19 languages listed here - I numbered them 1 through 4. The English word “animal” is listed as “B” for borrowed because this word did not come into modern English by descent from Old English, instead it was borrowed from French after the Norman invasion of England.

With this list of words we can calculate the similarity of Spanish, Portuguese, French and Italian as 100% because all five of the words are cognates with each other in these languages. The Romance languages share no cognates (in this set of only five words) with Hindi, one with Russian, and none with German. Dyen, Kruskal and Black (1992) counted the number of cognate words between languages in this way to estimate the phylogenetic relatedness of the 84 Indo-European languages. That is to say they collected a list of 200 words from each of 84 languages and for each word judged the cognate sets as I have illustrated for a small subset of only five words in table 6.1.

I will now use a small portion of the Dyen, Kruskal & Black (DKB) database to illustrate how one goes about cluster analysis. We'll do this by hand for this small example and then see how to perform cluster analysis in R.

Before turning to this example though, I want to make a pitch for using general purpose statistical software rather than discipline-specific software. Hierarchical clustering is a well-known procedure in statistics, though in evolutionary biology different names are given to the same procedures. For example, I think that the "single linkage" method described in this section is the same as the method called Unweighted Pair Group Method of Arithmetic averages (UPGMA) in the evolutionary biology literature, while the "average linkage" method is the method called by the biologists the Weighted Pair Group Method of Arithmetic averages (WPGMA). My preference is to use the more generally accepted terms (as with "logistic regression" instead of "varbrule" in chapter 5) so that our research is more closely tied with the state of the art in statistics and other sciences, and so that statistics manuals won't seem so disconnected from what we do in linguistic research. For example, identifying UPGMA and WPGMA with the single linkage and average linkage methods of hierarchical clustering opens the way to using general purpose statistical software like R in historical linguistics. This has many advantages including that (1) data entry and management is much easier and more convenient, (2) the general purpose software is less brittle with a wider base of users working out the kinks, (3) linguists use the same tools as other scientists, enhancing interdisciplinary possibilities, and (4) alternative data analysis methods may be suggested. On this last point I'm thinking, for example, of Ward's method of clustering, or the complete linkage approach. These clustering methods share features of single and average linkage and have not yet found their way into the biologist's toolbox (perhaps for good reason, perhaps not).

Now back to the example of cluster analysis. Table 6.2 shows a small portion of the DKB Celtic speech varieties "relatedness" matrix which was calcuated in terms of the percent cognates in their 200 words.

Table 6.2. Percent cognate words among seven Celtic speech varieties (Dyen, Kruskal & Black, 1992).

	Irish A	Irish B	Welsh N	Welsh C	Breton List	Breton SE	Breton ST
Irish A							
Irish B	82.6						
Welsh N	34.4	35.5					
Welsh C	34.2	36.0	93.9				
Breton List	31.1	33.5	63.9	61.5			
Breton SE	*30.9*	*32.1*	*62.9*	*61.0*	*88.8*		
Breton ST	**31.9**	**32.8**	**64.1**	**61.7**	**92.9**	94.9	

This set of similarity measures provides a nice way to demonstrate three key methods of clustering languages into a family tree. The tree given by hierarchical clustering using the "average linkage" model is shown in figure 6.2.

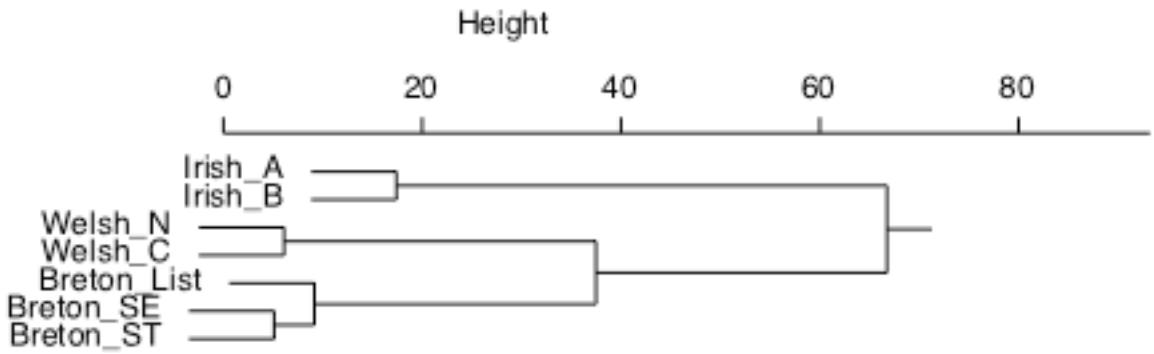

Figure 6.2. Results of hierarchical clustering of the data in table 6.2. Height, measured at nodes where clusters join each other, is 100 minus the percentage of cognates, so that the data reflect distance rather than similarity, and the linkage was done by averaging.

Perhaps the simplest hierarchical clustering algorithm is called single linkage. The idea is to join items into clusters on the basis of the similarity of a to-be-added item and the most similar member of the cluster. There are three steps.

Single linkage.

(1) find two most similar items
(2) link them into a cluster and collapse the rows and columns for the two items. If one is in an existing cluster add the new item into that cluster.
(3) repeat steps 1 and 2 until all items belong to a cluster.

So, for example, in table 6.2 Breton SE and Breton ST are the most similar two languages because they have the highest percentage of cognates (94.9%). So we join them into a cluster and measure the similarity of that cluster with the remaining languages in terms of the highest similarity between either Breton SE and Breton ST. This means that we can keep the percent cognate values shown in bold face in table 6.2 and remove the values shown in italics. Now we have reduced the size of the similarity matrix by one row and one column (table 6.3). This single linkage method looks for only one good match between a member of the cluster and the other items to be clustered. It could have happened that Breton SE is very similar to Breton List, while Breton ST is not similar to Breton List at all. In single linkage we ignore the mismatch between Breton ST and Breton List and base our decision about whether or not to add Breton List to Cluster 1 on the basis of the similarity of Breton SE and Breton List. As we will see, other clustering methods do not ignore mismatches.

Table 6.3. Collapsing table 6.2 by one row and one column (using single linkage).

	Irish A	Irish B	Welsh N	Welsh C	Breton List	Cluster 1
Irish A						
Irish B	82.6					
Welsh N	**34.4**	*35.5*				
Welsh C	*34.2*	**36.0**	93.9			
Breton List	31.1	33.5	**63.9**	*61.5*		
Cluster 1	31.9	32.8	**64.1**	*61.7*	92.9	

Now, we return to step (1) and search for the two most similar items in the matrix. This time Welsh N and Welsh C are most similar, and following the single linkage rule for collapsing rows and columns we keep the bold face items and delete the italicized similarity measures. This gives us table 6.4.

In table 6.4, the most similar items are Breton List and Cluster 1, so we add Breton List to Cluster 1 and collapse their rows keeping the bold-face highest similarities and deleting the

italicized lowest similarities. The result is in table 6.5.

Table 6.4. Collapsing table 6.3 by one row and one column (using single linkage).

	Irish A	Irish B	Cluster 2	Breton List	Cluster 1
Irish A					
Irish B	82.6				
Cluster 2	34.4	36.0			
Breton List	*31.1*	**33.5**	*63.9*		
Cluster 1	**31.9**	*32.8*	**64.1**	92.9	

Table 6.5. Collapsing table 6.4 by one row and one column (using single linkage).

	Irish A	Irish B	Cluster 2	Cluster 1
Irish A				
Irish B	82.6			
Cluster 2	*34.4*	**36.0**		
Cluster 1	*31.9*	**33.5**	64.1	

Irish A and B will be joined together into a third cluster because they have the highest similarity, and then cluster 1 and cluster 2 will be joined on the basis of the 64% cognation between Welsh N and Breton List, and finally cluster 3 (the Irish varieties) will be joined to the Breton/Welsh varieties on the basis of the 36% cognatives between Irish B and Welsh C.

The two other most commonly used hierarchical clustering methods are almost identical to this one, except that in "complete linkage" we keep the italicized similarities instead of the bold-faced ones, and in "average linkage" we keep the average of the italicized and bold-face similarities. The two extreme methods (single linkage and complete linkage), ignore either the lower or higher similarities and tend to be a little less stable than average linkage, so in the analyses presented below I used average linkage. Dyen et al. also indicated that, in their method of clustering, average linkage was used. For more details about cluster analysis, measures of similarity, and very helpful practical suggestions, see Aldendorfer & Blashfield (1984).

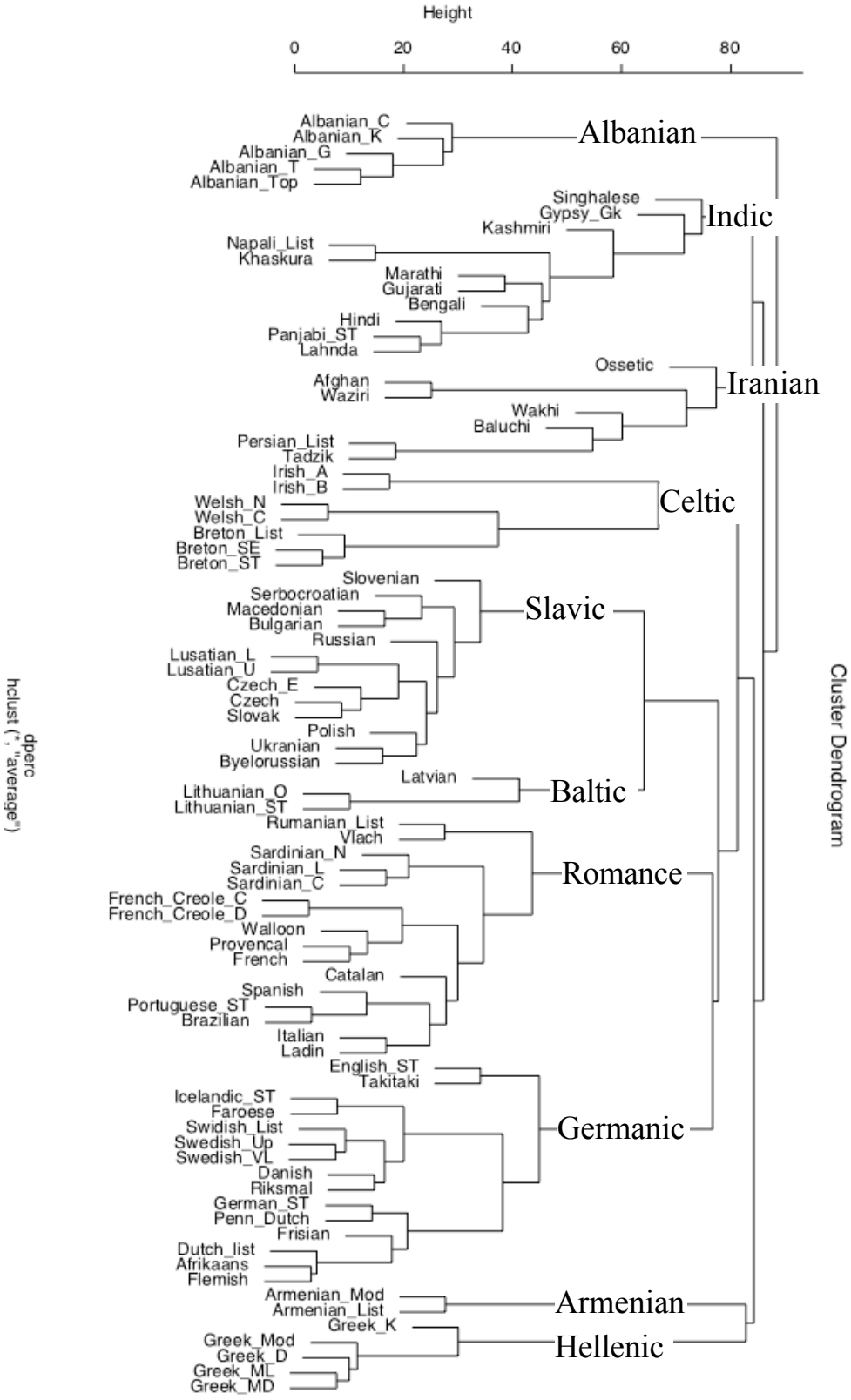
Height
0
20
40
60
80
Albanian_C
Albanian_K
Albanian_G
Albanian_T
Albanian_Top
Albanian
Singhalese
Gypsy_Gk
Kashmiri
Indic
Napali_List
Khaskura
Marathi
Gujarati
Bengali
Hindi
Panjabi_ST
Lahnda
Ossetic
Afghan
Waziri
Iranian
Wakhi
Baluchi
Persian_List
Tadzik
Irish_A
Irish_B
Celtic
Welsh_N
Welsh_C
Breton_List
Breton_SE
Breton_ST
Slovenian
Serbocroatian
Macedonian
Bulgarian
Slavic
Russian
Lusatian_L
Lusatian_U
Czech_E
Czech
Slovak
Polish
Ukranian
Byelorussian
Latvian
Lithuanian_O
Lithuanian_ST
Baltic
Rumanian_List
Vlach
Sardinian_N
Sardinian_L
Sardinian_C
Romance
French_Creole_C
French_Creole_D
Walloon
Provencal
French
Catalan
Spanish
Portuguese_ST
Brazilian
Italian
Ladin
English_ST
Takitaki
Icelandic_ST
Faroese
Swidish_List
Swedish_Up
Swedish_VL
Danish
Riksmal
German_ST
Penn_Dutch
Frisian
Dutch_list
Afrikaans
Flemish
Germanic
Armenian_Mod
Armenian_List
Armenian
Greek_K
Greek_Mod
Greek_D
Greek_ML
Greek_MD
Hellenic
dperc
hclust (*, "average")
Cluster Dendrogram

Figure 6.3. The full cluster solution for the DKB percent cognates data, plotted as a "dendogram".

As seen in Figure 6.3, cluster analysis of the DKB cognates data produces a family tree that is very sensible from a historical perspective. All of the major subfamilies of Indoeuropean are found (when representatives were available in the dataset) and the further subgroupings are interesting and possibly correct. Nakhleh et al. (n.d.), however, compared single linkage clustering with the cladistic methods that I will discuss in the next section and concluded single linkage was "clearly inferior".

R note. DKB's percent cognates data set is in "IE-perc84.txt". To produce the cluster analysis presented in this section I used the following R commands.

Read it as a matrix.

```
> m <- read.table("IE-perc84.txt")
```

This is a matrix of similarity measures, and the hclust() algorithm wants to have distance measures. Also I wanted to present the data in percentages, while the DKB table is in 1/10 of a percent, so I converted the values by applying a function to the table that subtracts the percent cognates values in the table from 1000 and divides by 10. The c(1,2) says to apply this to both the rows and columns of the table. as.dist() converts the matrix into a distance matrix object which is expected by hclust().

```
> subtr <- function(x) c((1000-x)/10)
> d <- as.dist(apply(m[1:84],c(1,2),subtr))
```

Now for a pretty print out we add names to the distance matrix. This was stored as column 85 of the input matrix in IE-perc84.txt.

```
> names(d) <- m$V85
```

Finally, we use average linkage in the clustering algorithm and plot the resulting object, which because it is a cluster object plots as a hierarchical tree.

```
> clus <- hclust(d,"ave")
> plot(clus)
```

6.3. Cladistic analysis: Combining character-based subtrees

An important alternative to cluster analysis for the determination of historical family trees comes to linguistics from evolutionary biology. Biologists have worked out a set of methods to reconstruct family trees of species on the basis of shared inherited traits and this "cladistic" analysis is now also being used in historical linguistics. In this section I will explain the cladistic approach and the assumptions that it involves, and then give a specific example using a portion of the Dyen Indo-European data set. The main points of this section are that cladistics is not hard to understand and the software is easy to download and use.

R note. That's right, here finally, we have found something that isn't implemented yet in R. However, the "ape" library does have some utility functions that are evidently useful to evolutionary biologists, and I guess this is a good time to mention that there are hundreds of contributed packages that contain libraries of useful R functions. Well, some are more useful than others and the R development team includes the most useful and bug-free in the standard distribution of R. The fact that some evolutionary biologists have been adding functions to a specialized phylogenetic analysis package suggests that in the not too distant future you will be able to peform cladistic analysis in R.

Let's say that when languages share a cognate word they share a heritable "character". In biology, characters may be things like "has legs", or "is warm blooded". In linguistics we may consider characters that are based on the lexicon such as "the word for 'animal' decends from a proto form [dyr]", but may also include things like "has the 'ruki' rule", or "underwent Grimm's law". I won't say much about how one determines the characters that go into the cladistic method, except to say that, as with all quantitative methods, the data that go into the analysis determine its success. Phylogenetic trees produced by computer programs can be just as wrong as trees produced by hand; it depends on the quality of the data. I'll be illustrating this principle further in this section, when I show you "family trees" of Indo-European that are obviously wrong. One key thing to avoid in constructing a data set for cladistic analysis is to avoid using language "characters" that could easily develop independently in different languages. For example, basic clausal word order is probably not a good character to use in grouping languages because SOV languages have developed into SVO independently in different language families. Cognate words are good because analysis of sound change can identify words that are similar in two languages because of common descent - clearly indicating a historical genetic relation between the languages.

Figure 6.4 shows groups of Germanic languages that we can identify on the basis of the words for

"animal", "bark (of tree)", and "to breathe" from table 6.1. We can consider these groupings to be subtrees of a single larger tree and consider ways to join the trees with each other.

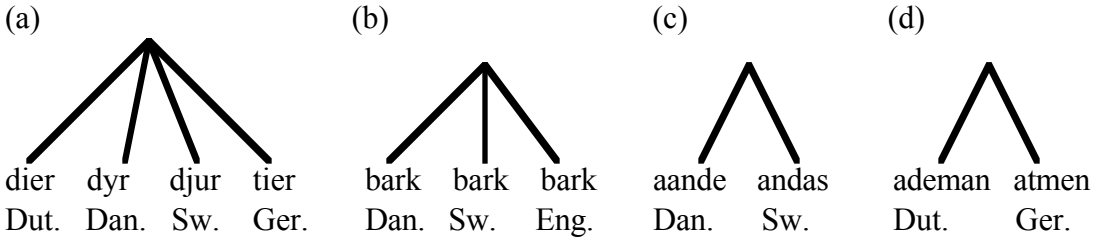

Figure 6.4. Germanic language groupings suggested by cognates for the words "animal", "bark (of a tree)" and "to breathe" in table 6.1. (Dut.=Dutch, Dan. = Danish, Sw. = Swedish, Ger. = German, Eng. = English).

For example, if we start with the largest group (a) and look at the two groups that are embedded in it, namely trees (c) and (d), we come up with the tree shown in figure 6.5 (a). There is a node that encompases Dutch and German (group d), a node that encompases Danish and Swedish (group c) and a node that encompases all four languages (group a). Unfortunately, though, there is no node that groups together Danish, Swedish, and English as is suggested by the word for "bark". The tree in figure 6.5 (b) shows a configuration that does have a node for group (b). However, note now that group (a) is not represented.

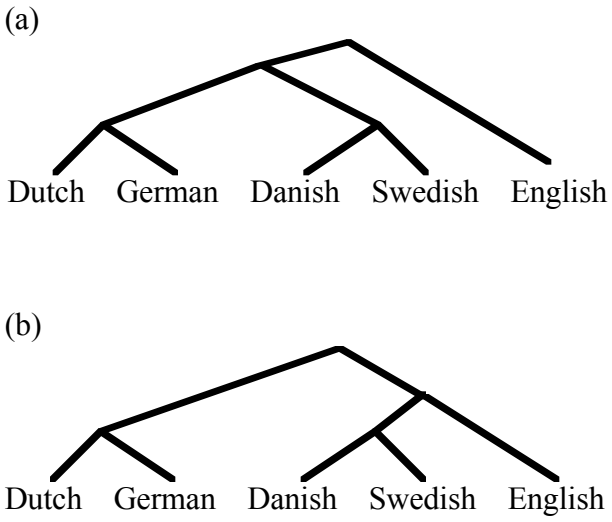

Figure 6.5. Two phylogenetic trees that are suggested by the groupings in figure 6.4.

This short illustration gives a flavor of the subtree combining method and illustrates the main problem in cladistics. Real data almost never define a single best-fitting tree, thus the method must choose from among many possible trees. Well, actually the method must be smart enough to generate lots of different trees that are all fairly consistent with the set of characters, and then measure how good the trees fit the data, and from a set of equally good trees choose a consensus tree that best represents the set of equally good trees.

Generating lots of different trees involves repeating the process of generating a tree from a number of different starting points (note how I suggested we start from group (a) or group (b) above), and adds nodes to the tree in different random orders. The options are fairly limited in the simple example in figures 6.4 and 6.5 but you can probably imagine how complicated the picture can become as we add hundreds of characters and dramatically increase the number of languages in the set as we would in a real research question.

When it comes to measuring how well a tree fits the data one of the most successful solutions is a maximum compatibility criterion that measures the success of a tree in term of the number of subtrees that are compatible with the overall tree. For example, both of the trees in figure 6.5 are compatible with three of the four subtrees in figure 6.4. The tree in figure 6.45(a) is incompatible with 6.4(b) and the tree in figure 6.5 (b) is incompatible with 6.4(a). This means that they are equally compatible with the subtrees suggested by the characters we derived from the words for "animal", "bark", and "to breathe". Two decide between these two options we need data from more words, or we need to put different weights on some of the characters so that some kinds of incompatibility are more costly than others. For example, if I were to decide that the weight of a character should be proportional to the number of languages that share that character (warning, I have no good reason for supposing this) then tree 6.5(a) wins because character 6.4 (a) is shared by four languages and 6.4 (b) is shared by only three. In a more realistic example of character weighting, Nakhleh et al. (2005) gave larger weights to phonological or morphological characters (like Grimm's law and the RUKI rule) than they did to lexical cognates. This is a way of allowing for the possibility that some of the cognate decisions may have been wrong.

To me the maximum compatibility measure is a pretty intuitive measure of how well a tree accounts for character data. It also seems to be the measure of choice in recent linguistic research (Nakhleh, 2005). Interestingly though, in evolutionary biology a different method - the maximum parsimony criterion - is the most common method of measuring the goodness of a phylogenetic tree, and most software packages are designed to optimize parsimony rather than the compatibility. In practice, the maximum parsimony method (see below) is used to generate a set of candidate trees which can then be evaluated by the maximum compatibility method.

Figure 6.6 illustrates how maximum parsimony is calcuated for the two candidate trees of

Germanic that we saw in figure 6.5. The trees have now been labeled with letters that correspond to the characters (that is, the subtrees) shown in figure 6.4. For example, both trees have a hatch mark labelled (d) on the branch that leads to Dutch and German. This signifies that Dutch and German "developed" character (d) - a cognate word for "to breathe" that is different from the word "to breathe" in the other languages. This concept of developing a character, or sharing a character state change is obviously relevant for physical characters like "has legs" but can also be applied to linguistic characters as well (perhaps more naturally for sound changes like "underwent the RUKI rule"). So, each labelled hatch mark on the tree corresponds to the development of a character. Notice that in figure 6.6(a) character (a) appears on the tree in two places - on the branch leading to Dutch and German, and on the branch leading to Danish and Swedish. The most parsimonious tree will have each character listed in only one place, so we can measure the parsimony of the tree by simply counting the number of hatch marks on it.

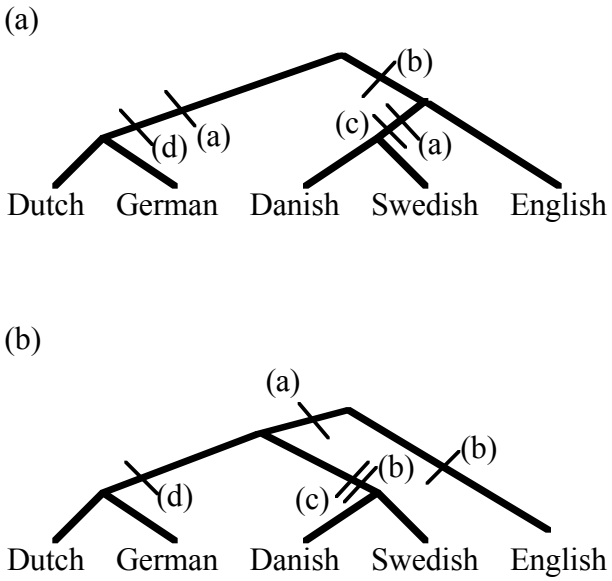

Figure 6.6. Two possible trees of Germanic languages showing character-state changes for calculation of the maximum parsimony goodness criterion.

Tree 6.6 (b) is more parsimonious than tree 6.6 (a) with respect to character (a), but now character (b) has to be marked at two places on the tree. The net result is that the two trees are equally parsimonious just as they are equally compatible with the set of subtrees defined by the character set.

Now, to conclude this section I'd like to walk you through an analysis using a parsimony-criterion optimization method for finding phylogenetic trees. The analysis is carried out using the software package Phylip which was developed by Joseph Felsenstein. This package is one

of the two most frequently cited in the evolutionary biology literature and unlike the other "Paup" is available for a free download (see http://evolution.gs.washington.edu/phylip.html).

The data that we'll analyze is again drawn from Dyen et al.'s Indo-European cognates data. Table 6.6 shows the cognates data that were used as input to the analysis. You may recognize the first six columns of data as coming from table 6.1. That is, the column that reads 1,1,1,1,B,2,2, etc. codes the cognates among words for "animal". The second column (beginning 3, 1,1,2,1, etc. codes the cognates among words for "bark (of a tree)", and so on. Table 6.1 has only five words in it and for this analysis I added nine more words so it would be a little less of a toy problem. The format of the first column - the list of names - is very strict. There must be exactly ten characters in this line. The first line is also needed by the Phylip software. This line specifies that there are 19 rows and 14 columns of data (not counting the names).

Table 6.6. A subset of the Dyen et al. (1992) Indo-European cognates data. This data is formated for input to the Phylip "pars" program.

```
 19 14
Dutch     1 3 4 1 2 3 3 2 1 2 3 1 1 2
Danish    1 1 4 1 5 3 3 2 1 4 3 1 1 2
Swedish   1 1 4 1 5 3 3 2 1 4 3 1 1 2
German    1 2 4 1 2 3 3 2 1 2 3 1 1 2
English   B 1 B 6 B 3 3 B 1 2 3 1 1 2
Slovenian 2 3 5 3 1 2 3 2 1 3 1 2 4 3
Russian   2 3 5 3 1 2 3 2 2 3 1 2 4 3
Polish    2 3 5 3 1 2 3 2 1 3 2 2 4 3
Czech     2 3 5 3 1 2 3 2 1 3 1 2 4 3
Greek     2 6 1 7 3 1 3 2 1 5 3 2 7 6
Kashmiri  3 B 3 2 4 2 2 B 1 B 3 2 5 4
Gujarati  3 4 3 2 4 2 2 1 1 3 B B 5 B
Marathi   3 4 3 2 4 2 2 1 B 3 3 2 6 4
Hindi     3 4 3 2 4 2 2 1 1 3 3 2 5 4
Waziri    3 5 6 5 4 4 1 2 3 6 4 2 2 5
Portuguese4 3 2 4 6 2 3 2 1 1 3 2 3 1
Spanish   4 3 2 4 6 2 3 2 1 1 3 2 3 1
Italian   4 3 2 4 6 2 3 2 1 1 3 2 3 1
French    4 3 2 4 6 2 3 2 1 1 3 2 3 1
```

To construct a phylogenetic tree that shows the linguistic family tree (figures 6.7 and 6.8) I used a sequence of three programs in the Phylip package (these are discussed in more detail in the "Phylip note" below). As we saw above, it is usually the case that there is no one best tree to account for all of the data whether we use the maximum compatibility method or the maximum

parsimony method, so the usual method in cladistic analysis is to produce a number of equally good trees and then construct a consensus tree that retains clusters that are in most of the "good" trees.

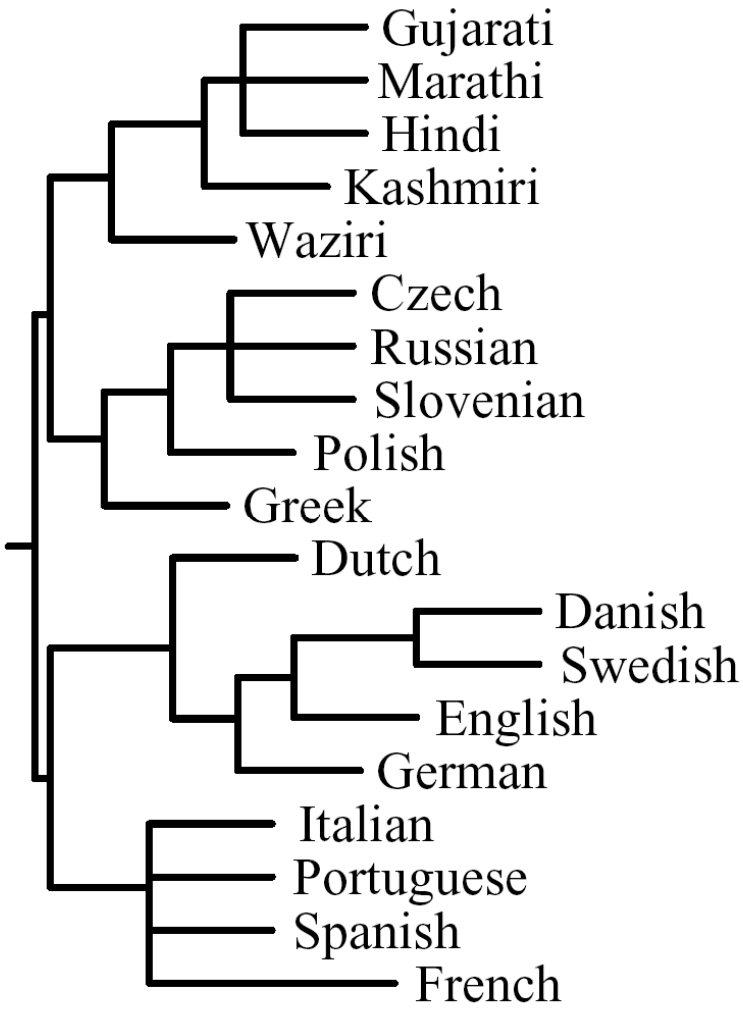

Figure 6.7. The consensus tree produced by maximum parsimony cladistic analysis of the data in table 6.6. The tree is rooted from its midpoint, with about as many languages on either branch from the root. Line length indicates the number of characters that distinguish the cluster or language linked by the line.

In order to explore the (very large) space of possible good trees for this data set I had the program randomize the order in which the languages would be considered differently for each new tree, and to keep as many as 100 different trees that tied for best. Then I used a majority rule

method to select a tree that retains groupings that are present in most of the candidate trees. Interestingly, this process results in the tree shown in figure 6.8, with no root node. Selection of a root node is actually sort of arbitrary. You can imagine picking the tree in figure 6.8 up by any node, which then becomes the root. For figure 6.7 I used a third program to "retree" the tree assigning a root node using the midpoint method.

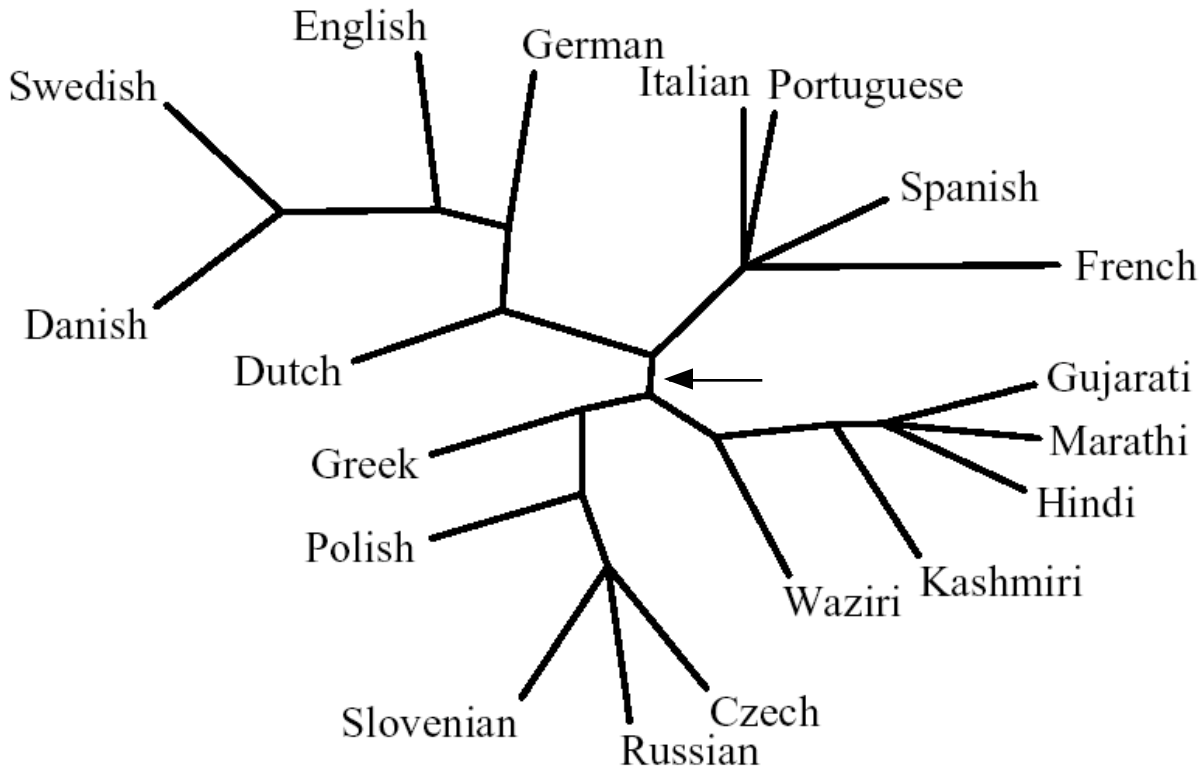

Figure 6.8. The same tree as in figure 6.7, but with no root assigned. The arrow marks the link that was chosen for the root in figure 6.7.

Phylip note - In this section I used programs in the Phylogeny Inference Package (Phylip) by Joseph Felsenstein (see http://evolution.gs.washington.edu/phylip.html to download it). I stored the data set that was shown in table 6.5 in a file called "dyen_subset_coded.txt" and used that as input to the `pars` program. `pars` uses the Wagner parsimony method to find maximum parsimony phylogenetic trees. I used the "J" option to randomize the input order of the data differently for different runs so that a range of possible trees would be generated. The program saves all of the trees (up to 100 by default) that are tied for best. The output of `pars` is then fed into the program `consense` to compute one tree that captures most of the structure that is

shared by the trees found by `pars`. Finally, the `retree` program is used to add a root to the consensus tree and to adjust the orientation of the leaves of the tree so that it looks good. Finally, not shown here, there is a program to draw a nice picture using the tree file created by retree.

In general, you must interact with these programs in a rather clunky text interface. The problem with this is not that it is a text interface, but that you can't save a set of commands for later reference or reuse. You basically have to remember what you did and go through the entire procedure again next time. Also, all input files, such as dyen_subset_coded.txt, must be copied into the phylip folder and output files are automatically written in that folder. Here are what the interfaces of each of these programs look like.

---------- Interacting with the **pars** program ----------

```
Pars: can't find input file "infile"
Please enter a new file name> dyen_subset_coded.txt

Pars: the file "outfile" that you wanted to
     use as output file already exists.
     Do you want to Replace it, Append to it,
     write to a new File, or Quit?
     (please type R, A, F, or Q)
r

Discrete character parsimony algorithm, version 3.65

Setting for this run:
  U                 Search for best tree?  Yes
  S                       Search option?  More thorough search
  V              Number of trees to save?  100
  J     Randomize input order of species?  No. Use input order
  O                        Outgroup root?  No, use as outgroup species 1
  T              Use Threshold parsimony?  No, use ordinary parsimony
  W                        Sites weighted?  No
  M           Analyze multiple data sets?  No
  I            Input species interleaved?  Yes
  0   Terminal type (IBM PC, ANSI, none)?  (none)
  1    Print out the data at start of run  No
  2  Print indications of progress of run  Yes
  3                        Print out tree  Yes
  4          Print out steps in each site  No
  5  Print character at all nodes of tree  No
  6       Write out trees onto tree file?  Yes

  Y to accept these or type the letter for one to change
```

---------- Interacting with the **consense** program ----------

```
Consense: can't find input tree file "intree"
Please enter a new file name> parstree

Consense: the file "outfile" that you wanted to
     use as output file already exists.
     Do you want to Replace it, Append to it,
     write to a new File, or Quit?
     (please type R, A, F, or Q)
f
Please enter a new file name> consenseout.txt

Consensus tree program, version 3.65

Settings for this run:
 C         Consensus type (MRe, strict, MR, Ml):  Majority rule (extended)
 O                              Outgroup root:  No, outgroup species  1
 R                 Trees to be treated as Rooted:  No
 T             Terminal type (IBM PC, ANSI, none):  (none)
 1                  Print out the sets of species:  Yes
 2         Print indications of progress of run:  Yes
 3                             Print out tree:  Yes
 4                Write out trees onto tree file:  Yes

Are these settings correct? (type Y or the letter for one to change)
```

---------- Interacting with the **retree** program ----------

```
Tree Rearrangement, version 3.65

Settings for this run:
  U          Initial tree (arbitrary, user, specify)?  User tree from file
  N   Format to write out trees (PHYLIP, Nexus, XML)?  PHYLIP
  0                      Graphics type (IBM PC, ANSI)?  (none)
  W       Width of terminal screen, of plotting area?  80, 80
  L                         Number of lines on screen?  24

Are these settings correct? (type Y or the letter for one to change)
------------------------------
```

The cladistic method for combining character-based trees is a pretty intuitively simple one, and does yield sensible trees. However, the results are not stable depending on luck of the draw in

the random orderings that get tested, and more importantly on the accuracy of particular judgements that the linguist makes in establishing the character sets used as input to the process. In fact Nakhleh et al. (nd) conclude their methodological study with the sobering suggestion that "it seems possible that phylogenetic reconstruction methods are best suited to working out, in a maximally rigorous fashion, the consequences of linguist's judgements" (p. 22). This is a useful thing to do, but does indicate that cladistic analysis is not a magic method that will automatically deliver the phylogenetic tree of a language family.

6.4. Clustering on the basis of spelling similarity

There are some problems with cladistics that limit its utility in some circumstances. In order to use the method we have to know enough about the history of each language under investigation to be able to identify words as cognates. For comparisons at a fairly shallow time depth, or with a small number of languages, this is not a problem, but for more distant undocumented relationships we can't really know whether languages are similar because of borrowing or direct descent. Nakhleh et al. (2005) indicate this in their "phylogenetic networks" approach when they add links for borrowing in a tree determined on the basis of cognate words (among other things). It may be that no methodology is immune from this concern, but I would like to consider an additional technique here, as an additional tool for historical phonetic research.

Even for comparisons of relatively shallow time depth, in working with languages for which we don't have a body of documentary evidence we can't always know whether words are cognates with each other. Especially for non Indo-European languages it would be convenient to be able to explore the similarities among speech varieties even when historical information may not be available or reliable.

For example, Ladefoged, Glick & Criper (1971, p. 34) in their study of the languages of Uganda (now Tanzania) noted some cases of historical uncertainties in the movements and interactions among peoples. They suggested that, "Because of mixtures of this kind, and because we do not know enough about the past relations between the tribal groupings in Uganda, the historical method is not very useful in trying to make a detailed classification of the languages of Uganda". They introduced a technique for clustering languages on the basis of their phonetic similarities. This section presents an extension of their method.

I told Peter Ladefoged about this "new" clustering method that I was toying with. He was gracious about it but basically told me that he had also thought of exploring language similarities on the basis of phonetic comparisons of words - when I was in junior high school! One of the homework exercises at the end of this chapter involves constructing a map of Bantu languages using data from Ladefoged, Glick & Criper (1971).

To explore the utility of cluster analysis without historical reconstructions I used a dynammic time warping method to calculate the distances between words on the basis of their spellings and then entered these distances into a cluster analysis of Indo-European. I should say that I don't think that this takes the place of making cognate decisions on the basis of systematic sound changes. Instead, I am offering this method as an exploratory tool that may be valuable in the investigation of language prehistory.

The dynamic time warping method has been used to calculate the similarities of dialects in computational dialectology (Heeringa, 2004; see also Heeringa & Braun, 2003 and Heeringa & Gooskens, 2003; and especially Sankoff & Kruskal, 1983/1999), and in the analysis of phonetic reduction in conversational speech (Johnson, 2003). However, in both my earlier work and in Heeringa's work, the method calculates pronunciation differences using phonetic transcriptions, so in this study there was no guarantee that spelling similarity would produce a sensible analysis of the Indoeuropean language family. The reason that I decided to try this is that it is often very difficult to know whether two words are cognate with each other in two languages - especially if the languages are distantly related. Additionally, it might be of some interest to compare a historical relatedness tree with a tree of language similarity which takes into account borrowing as a source of similarity. This spelling similarity method, though very crude, makes the beginnings of such a comparison possible. Perhaps there is a way to measure how much of language similarity is due to "genetic" relatedness, and how much is due to chance similarity or borrowing.

In the spelling comparison algorithm, the spellings are aligned with each other allowing for insertion, deletion, and substitution of symbols, so that identical symbols will be aligned with each other and so that vowel symbols will tend to line up with each other. Three example spelling alignments are shown in table 6.6.

Table 6.6. Spelling "distance" produced by the dynamic time warping algorithm. The letters align with each other in columns according to the algorithm's choice for the best possible alignment of the spellings.

English	a	n	i	m	a	l	.	
Italian	a	n	i	m	a	l	e	0.115

English	.	a	n	i	m	a	l	
Kashmiri	j	a	n	a	w	a	r	0.499

English	.	a	.	.	.	n	i	m	a	l	
Russian	z	i	v	o	t	n	o	.	e	.	0.856

In this table, the only difference between English "animal" and Italian "animale" is the deletion of the final "e" in English (or insertion of "e" in Italian, if you want to think of English as the base). Vowel substitutions added 1 to the distance. Insertions and deletions added 1.5, and substitutions involving consonants added 2 to the distance. The reported distance is the average distance. For example, the total distance between the English and Italian words is 1.5 so the average is 1.5/(6+7) because there are six letters in "animal" and seven in "animale" (Heeringa, 2004, prefers to divide by the length of the longest string).

Now to come up with an overall measure similarity between English and Italian, for example, we take the average of the 200 word similarities. This value for English and Italian is 0.755. The speech variety that scored the most similar to English is Frisian with an average distance of 0.566, though in the cluster solution below Frisian groups with Afrikaans, Dutch and Flemish rather than with English because it is even more similar to these (particularly Dutch with a distance of only 0.391).

With a matrix of overall spelling similarities among the languages we can apply hierarchical clustering as we did with the percent cognates data. The results are shown in figure 6.9. What is interesting about these results is that the major language families are again found, despite the very crude methodology used to compare languages. Some of the differences between a family tree based on cognation (figure 6.3) and this one based only on spelling similarity must result from chance similarity, and perhaps also on the presence of borrowed word similarity. For instance, Albanian, a separate branch of Indo-European shows greater affinity for the Hellenic branch (from which it has borrowed many words) in this spelling similarity analysis.

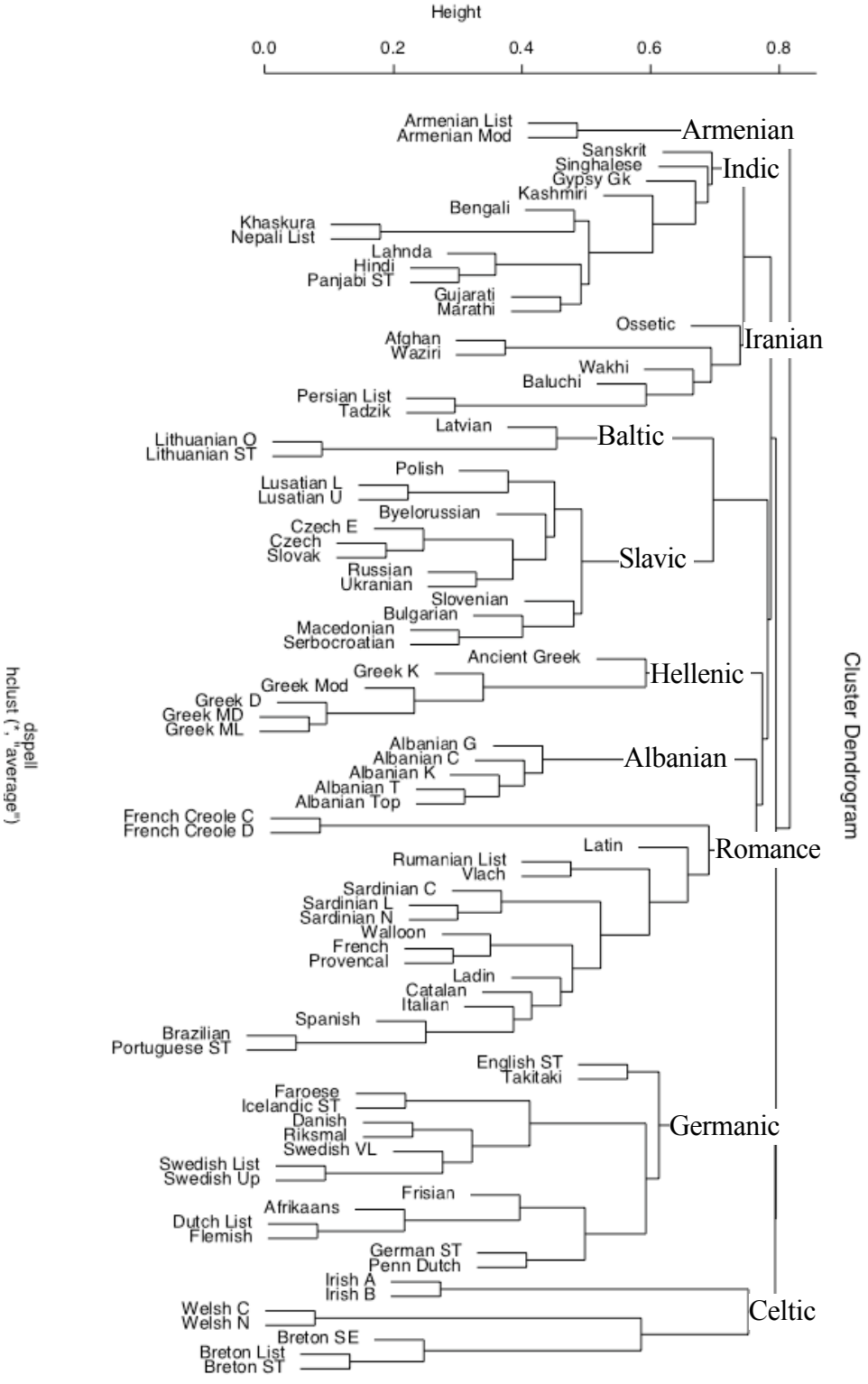
Height
0.0
0.2
0.4
0.6
0.8
Armenian List
Armenian Mod
Armenian
Sanskrit
Singhalese
Gypsy Gk
Indic
Kashmiri
Bengali
Khaskura
Nepali List
Lahnda
Hindi
Panjabi ST
Gujarati
Marathi
Ossetic
Afghan
Waziri
Iranian
Wakhi
Baluchi
Persian List
Tadzik
Latvian
Baltic
Lithuanian O
Lithuanian ST
Polish
Lusatian L
Lusatian U
Byelorussian
Czech E
Czech
Slovak
Russian
Ukranian
Slavic
Slovenian
Bulgarian
Macedonian
Serbocroatian
Ancient Greek
Greek K
Hellenic
Greek Mod
Greek D
Greek MD
Greek ML
Albanian G
Albanian C
Albanian K
Albanian T
Albanian Top
Albanian
French Creole C
French Creole D
Latin
Romance
Rumanian List
Vlach
Sardinian C
Sardinian L
Sardinian N
Walloon
French
Provencal
Ladin
Catalan
Italian
Spanish
Brazilian
Portuguese ST
English ST
Takitaki
Faroese
Icelandic ST
Danish
Riksmal
Swedish VL
Germanic
Swedish List
Swedish Up
Frisian
Afrikaans
Dutch List
Flemish
German ST
Penn Dutch
Irish A
Irish B
Welsh C
Welsh N
Celtic
Breton SE
Breton List
Breton ST
Cluster Dendrogram
dspell
hclust (*, "average")

Figure 6.9. Dendogram of clusters produced in a hierarchical clustering analysis of spelling distances.

You will notice in Figure 6.9 that I also added Sanskrit, Ancient Greek, and Latin word lists to the lists that were analyzed in Dyen et al. (1992). Interestingly, in the analysis of spelling similarities, these ancient languages cluster with their historical descendents, being the last to cluster with their family near the top of the tree. Of course you wouldn't want to make too much of this because creole languages, like French Creole, and languages that have many borrowed words, like English and Ossetic, also cluster more weakly with their relatives. Nonetheless, it is very interesting that such a crude analysis of overall language similarity could produce such an informative picture of the language family.

Naturally, there is a caveat. It could be that much of the successful grouping found in figure 6.9 arises from the borrowing of writing systems, or more recently shared conventions of romanization. This analysis needs to be done on the basis of broad phonetic transcription rather than on orthography. Despite this concern, I'll mention a couple of other steps of analysis using the spelling similarity data.

R note. The perl script that I used to calculate the spelling distances between languages is "get_IE_distance". There are also a couple of support scripts, to condition the data into a distance matrix. You can get these on the book web page, and read the comments in the scripts to see what I did. The R commands to produce figure 6.9 are pretty much the same as those we used to produce figure 6.3.

```
> m <- read.table("IE-distmatrix.txt",sep = ",")
> dspell <- as.dist(m[1:87])  # dyen's 84 languages + 3 ancient langs.
> names(dspell) <- m$V88
> plot(hclust(dspell,"ave"))
```

Perhaps one could measure the impact of borrowing in the spelling similarity analysis by correlating genetic distance from Dyen et al.'s (1992) "percent cognates" distance measure with this simple spelling distance measure. I did this by regressing the spelling distance measure against the percent cognates distance measure. As figure 6.10 shows, the relationship between these two measures of language similarity is non-linear so I regressed the log of cognate distance against the spelling distance. The R^2 value gives a proportion of genetic similarity that is accounted for by spelling similarity. The value that I obtained for this R^2 value was 0.86, indicating that 86% of the variance in the cognate judgements can be predicted from the spellings of the words. The remaining 14% of variance is possibly contributed by borrowing.

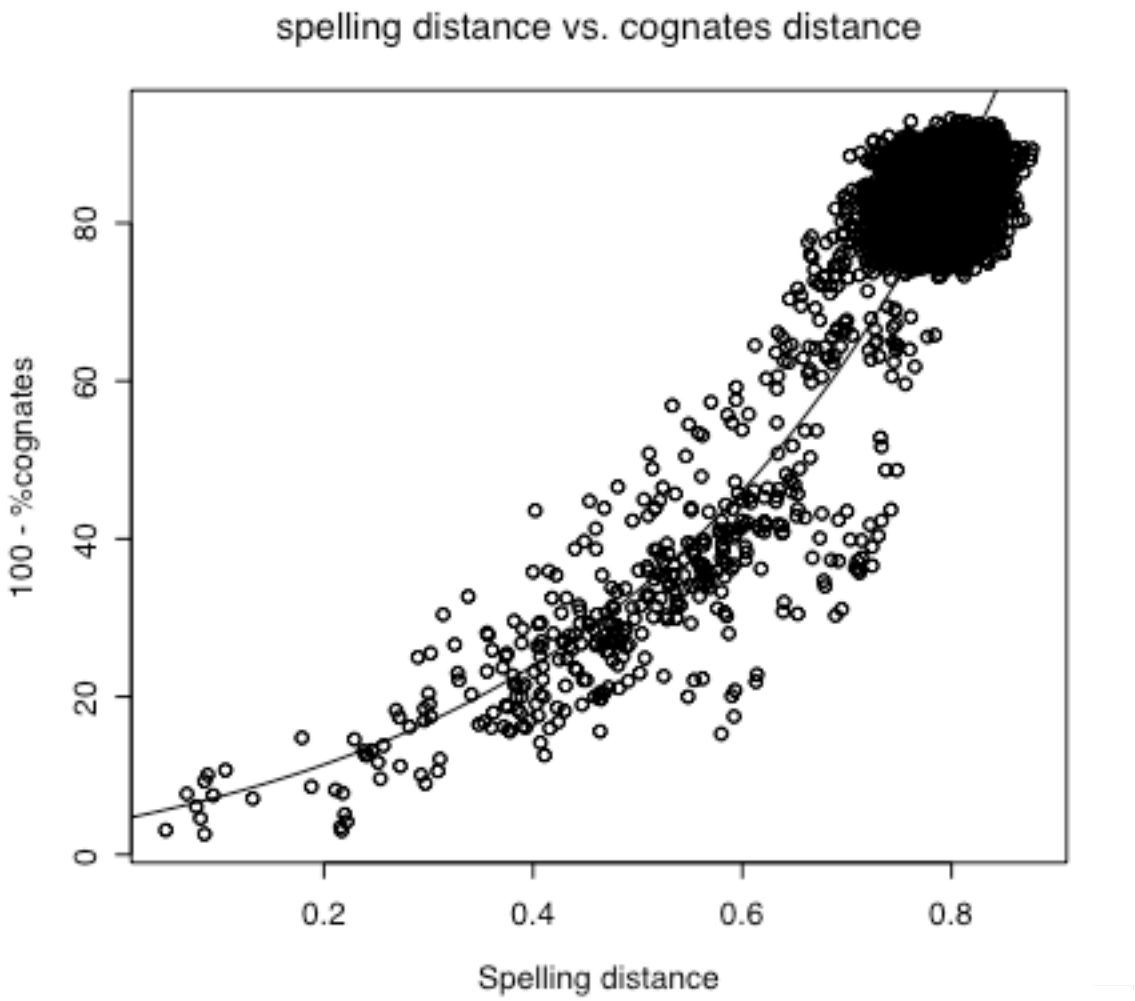

Figure 6.10. Spelling distance is correlated with genetic distance. Each dot in this graph is a language comparison representing the overall distance on the 200-word Swadesh list. Most languages in the data set share fewer than 25% cognate words (the clump of dots in the upper right of the graph). The regression line is for the formula: log(cognate_distance + 5) = 2.2299 + 2.83923 * spelling_distance.

R note. I converted the distance matricies that were read for the cluster analyses above into vectors using the command:

```
> vcog = c(m$V1,m$V2,.... m$V84)
```

I removed the diagonal of the matrix, where distance values were zero, and then did a regression

analysis.

```
> vcog2 = vcog[vcog>0]
> vspell2 = vspell[vspell>0]
```

I had looked at the graph and knew that the relationship was nonlinear so I did the regression on the log of the cognates distance (percent non-cognates) and found by trial and error that adding 5% (i.e. increasing the cognates distance slightly) and then taking the log, produced a slightly better fit to the data.

```
> summary(lm(log(vcog2+5)~vspell2))
Call: lm(formula = log(vcog2 + 5) ~ vspell2)
Residuals:
Min       1Q   Median       3Q      Max
-0.86603 -0.06250  0.00598  0.07110  0.51236
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.22990    0.01027   217.2   <2e-16 ***
vspell2      2.83923    0.01347   210.8   <2e-16 ***
--- Signif. codes:
0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
Residual standard error: 0.1201 on 6970 degrees of freedom
Multiple R-Squared: 0.8644,   Adjusted R-squared: 0.8644
F-statistic: 4.444e+04 on 1 and 6970 DF,  p-value: < 2.2e-16
```

Figure 6.10 was produced by these commands.

```
> plot(vspell2,vcog2,xlab="Spelling distance", ylab="100 - %cognates")
> curve(exp(2.2299+2.83923*x)-5,add=T)
```

6.5 Multidimensional Scaling - a language similarity space

Multidimensional scaling is a method for visualizing disimilarity data. The basic idea is that we can make a map using estimated distances between the points on the map by using a method of triangulation and iterative improvement between the data and the map. For example, if you know that the distance between Albany and Albuquerque is 2040 miles, and the distance between Albany and Atlanta is 1010 miles, and the distance between Albuquerque and Atlanta is 1400 miles, then you can construct a map showing the relative locations of these cities. Chapter 4 of my "Acoustic and Auditory Phonetics" has a fairly long non-technical discussion of MDS, and if you want to use MDS in any serious way you should study Kruskal and Wish (1978).

The spelling distance matrix can be used to plot a "similarity" space of languages using multidimensional scaling. This approach to the analysis of Indo-European might be valuable if we wish to explore similarities among the languages without assuming the genetic relationships as depicted in a family tree. Note that unlike principal components analysis, in order to perform multidimensional scaling we do not need to measure the languages on any variables - the only information we need is some measure of distance between languages, it could even be subjective judgements given by Indo-Europeanists if we wish!

The MDS solution for a three dimensional similarity space is produced using Kruskal's nonmetric MDS (see Kruskal and Wish, 1978) to map the language differences as measured by our spelling distance comparison.

Unlike a road map, in our map of Indo-European we don't know how many dimensions to include in the langauge similarity space so we try a number of maps and measure how well the map matches the data using a parameter called "stress" (Figure 6.11). This is a number on a scale from 0 (perfect fit) to 100 (the map captures nothing about the data). Generally, we are looking for a stress value less than 20 and an "elbow" in the stress by number of dimensions plot. The elbow indicates a point of dimenishing returns where adding new dimensions to the similarity space does not result in much increase in correspondence between the map distances and the raw data distances.

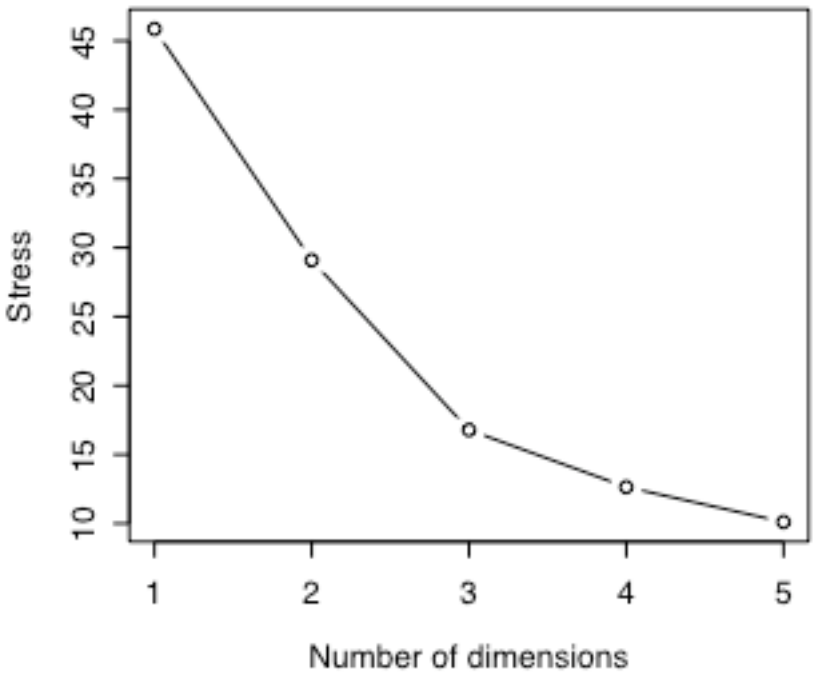

Figure 6.11. Stress of MDS solutions of 1 to 5 dimensions. Adding a fourth or fifth dimension does not substantially improve the over the 3 dimensional solution.

Figure 6.11 indicates that the best solution is probably the one with three-dimensions. This three dimensional solution is shown in detail in figures 6.12 and 6.13, and in a slightly less detailed way in figure 6.14. These views of Indo-European are complicated and rich, and perhaps informative about the historical and contemporary relations among the languages in ways that a cladistic view may not be capable of representing. Whether or not MDS will be of much use in historical linguistics is an open question, though the technique has been used with success in dialectology and language documentation (e.g. Ladefoged, Glick, an Criper, 1971).

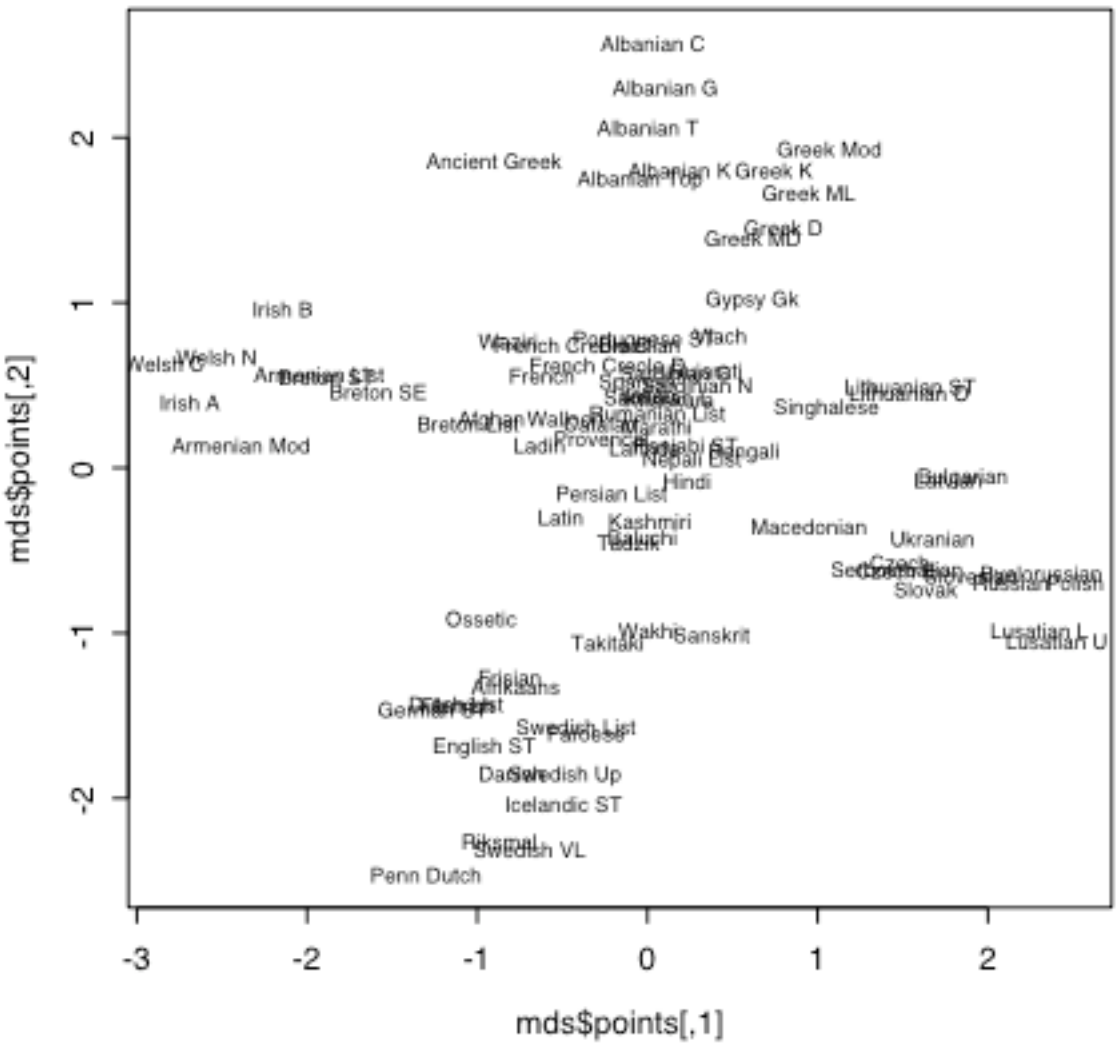

Figure 6.12. A plot of dimension one (horizontal axis) and dimension two (vertical axis) of the MDS similarity map of Indo-European speech varieties.

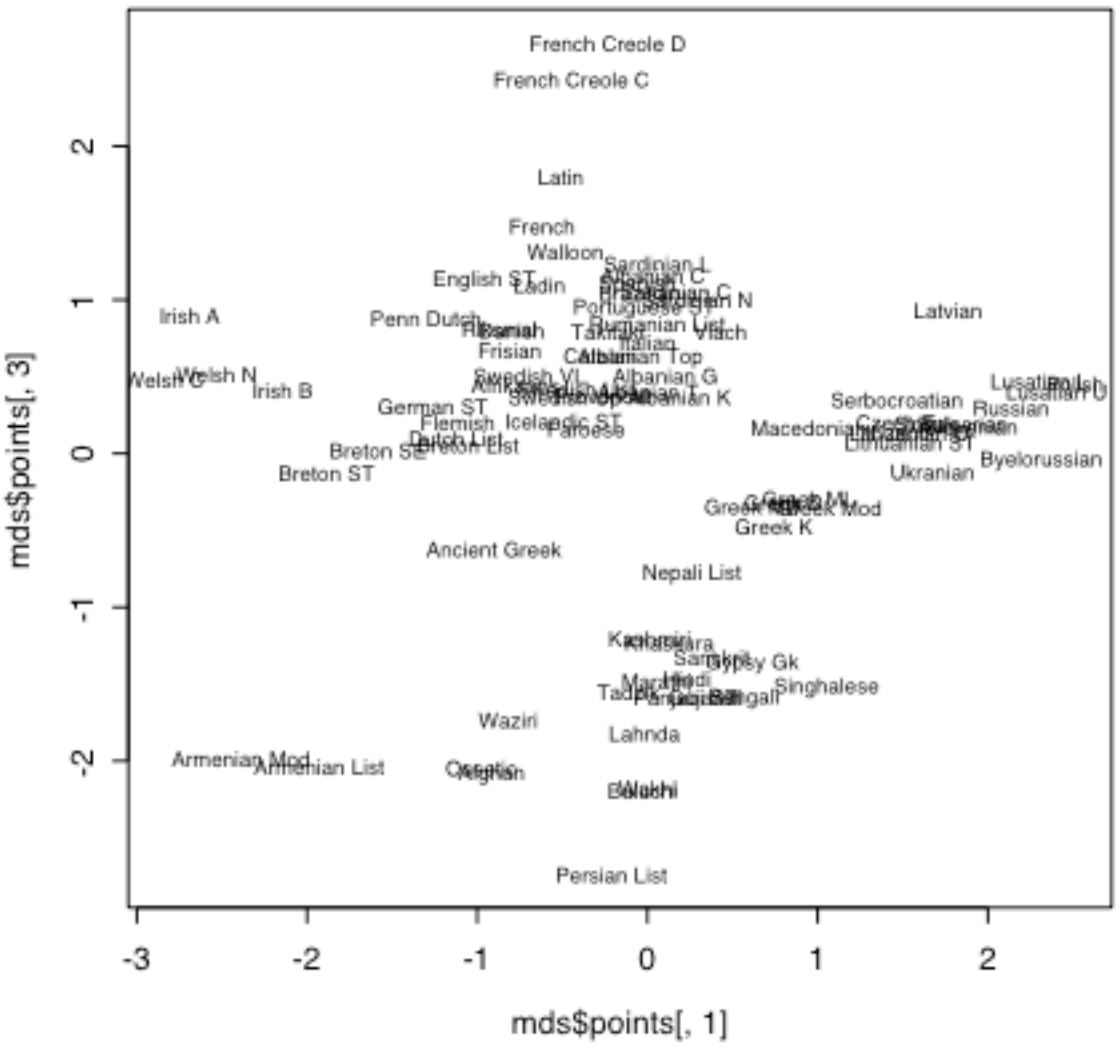

Figure 6.13. A plot of dimension one (horizontal axis) and dimension three (vertical axis) of the MDS similarity map of Indo-European speech varieties.

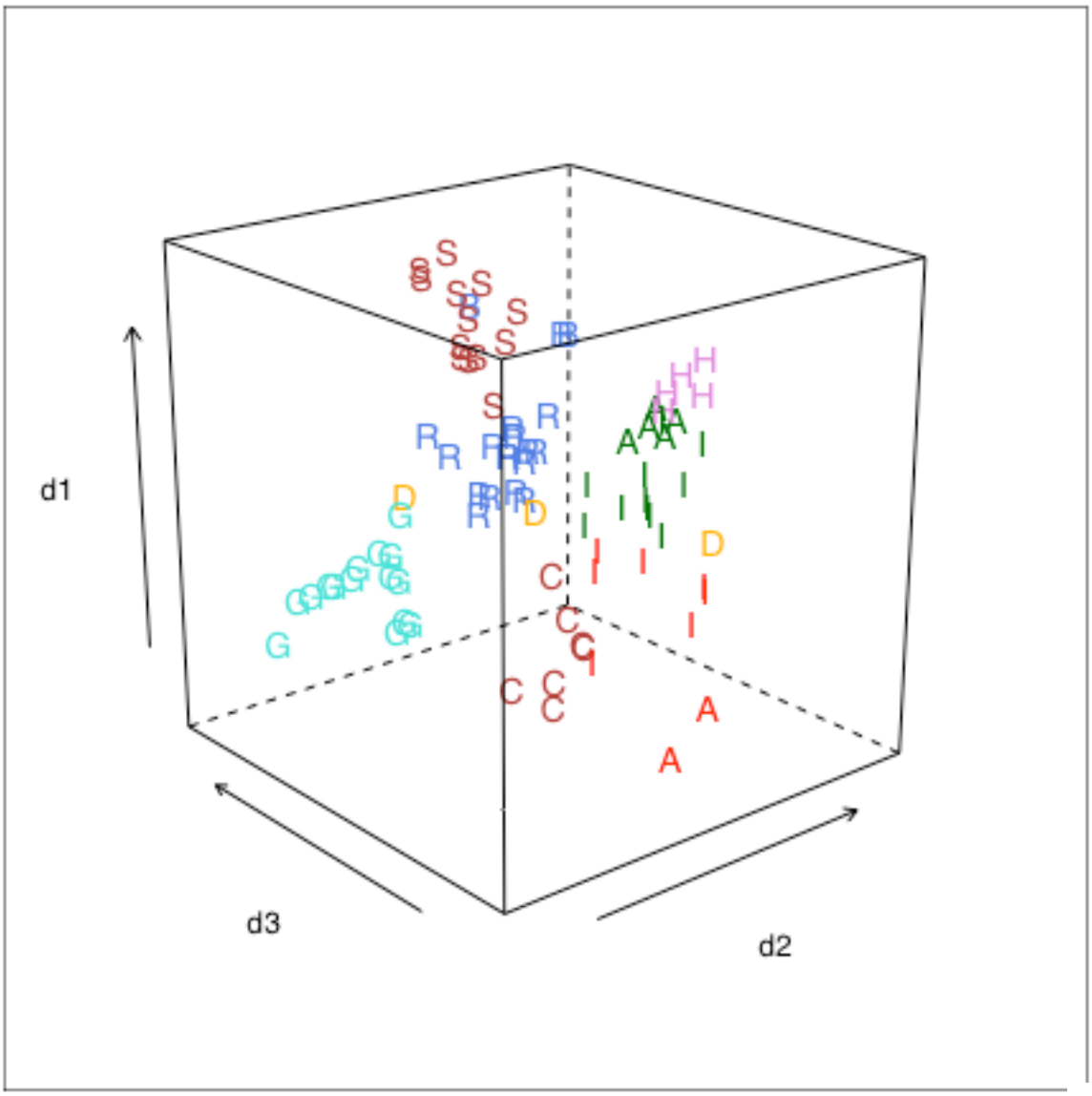

Figure 6.14. A plot of the three-dimensional similarity space of Indo-European languages, based on spelling similarity. Germanic languages are coded with "G", Hellenic with "H", Slavic with "S", Romance with "R", Baltic with "B", Celtic with "C", and the dead languages with "D". Indic and Iranian are both coded with "I", and Albanian and Armenian are both coded with "A" - Albanian is closer to Hellenic. The letters are also coded in different colors in the original.

R note. You can use the "metric" or "classical" multidimensional scaling procedure cmdscale() to

produce a map from city distances. For example, try out this code for a map of cities starting with the letter "A".

```
> m <- matrix(c(0,2040,1010,2040,0,1400,1010,1400,0),nrow=3,ncol=3,byrow=T,
dimnames=list(c("Albany","Albuquerque","Atlanta"),c("Albany","Albuquerque","At
lanta")))
> d <- as.dist(m)
> d
            Albany Albuquerque
Albuquerque   2040
Atlanta       1010        1400

> mds <- cmdscale(d)
> plot(mds,ylim=c(500,-300),xlim=c(-1500,1200),type="n")
> text(mds,as.character(dimnames(m)[[1]]))
```

The nonmetric, rank-order multidimensional scaling procedure that I used to visualize our language similarity data is implemented by the isoMDS() command, which is in the R library MASS. Here are commands to load that library and ask for a 2 dimensional solution for the dspell data set.

```
> library(MASS)
> m <- read.table("IE-distmatrix.txt",sep=",")
> d <- as.dist(m[1:87])
> names(d) <- m$V88
> mds <- isoMDS(d,k=3)
initial  value 37.282523
iter   5 value 22.490628
iter  10 value 17.671007
iter  15 value 17.073035
final  value 16.780951
converged
```

The final "stress" value measures the degree of correspondence between distances in the data and distances in the map and is given on a scale from 0 to 100 (in some implementations the stress scale is 0 to 1). As the fit improves the stress value approaches zero, and generally fits that have stress greater than 15-20 are considered unstable and not to be trusted. The the 3D solution represents a 12 point improvement over the 2D solution (stress = 29 versus stress = 16.8) and so is presented here.

```
> stress <- c(45.86,29.09,16.78,12.64,10.11)
```

```
> dim <- c(1,2,3,4,5)
> plot(stress,dim,type=b,xlab="Number of dimensions",
ylab="Stress")
```

The two figures showing the MDS similarity space (figures 6.12 and 6.13) were created with:

```
> plot(mds$points,type="n")
> text(mds$points,as.character(names(d)),cex=0.5)
> plot(mds$points[,1],mds$points[,3],type="n")
> text(mds$points[,1],mds$points[,3],as.character(names(d)),
cex=0.5)
```

I made a data frame that I called "space" to hold the language names, and the mds$points columns from the multidimensonal scaling solution. I also added a column that identifies the language family of each language, as revealed in the cluster analysis - except that I created a group called "DEAD" that includes Latin, Ancient Greek, and Sanskit so we can more easily see where they fall in the three dimensional similarity space.

```
> space <- read.delim("3dspace.txt")
> space
          language    group           d1          d2          d3
1           Afghan  Iranian -0.9160043711  0.28414069 -2.08811418
2        Afrikaans Germanic -0.7752611315 -1.32240609  0.44734323
3       Albanian C Albanian  0.0338698088  2.57451648  1.16025233
4       Albanian G Albanian  0.1064020998  2.30977783  0.51065756
```

To produce the 3D graph in figure 6.14, I used the "lattice" library of graphics commands (steeper learning curve than the other plotting routines in R, but fancier graphs).

```
> library(lattice)
> lset(col.whitebg())
> lset(list(superpose.symbol=list(pch
      =c("A","A","B","C","D","G","H","I","I","R","S"))))
> cloud(d1 ~ d2*d3,groups=group,data=space,cex=1.7,screen=list(z=40,x= -70))
---------------------------
```

Exercises

1. File "bantu_matrix.txt" contains a distance matrix for eight Bantu languages from the Tanzanian Language Survey (http://www.cbold.ddl.ish-lyon.cnrs.fr/Docs/TLS.html). I selected

33 words in the survey and judged whether the languages seemed to have the same stem for the word. The distance value in this table then is the number of times the two languages had a different stem out of 33. Construct, by hand, a hierarchical clustering tree of the languages using the single-linkage method, just as I did for the Celtic languages in section 6.2 of this chapter.

2. Using the distance matrix in file "bantu_matrix.txt". of Bantu languages in Tanzania. Perform hierarchical cluster analyses using average linkage, complete linkage, and Ward's method. How do these differ from each other and how do they differ from the single linkage tree you drew by hand for question 1. Explain any differences that you find.

3. File "LGC_phonetic.txt" contains phonetic distance data from Ladefoged, Glick, and Criper (1971). These data contrast Bantu languages in Tanzania according to how phonetically different they are for a set of twenty-two words. Phonetic differences were defined in terms of the number of phonetic features the stems of the words shared. Produce a hierarchical analysis of the data.

4. File "bantu_pars.txt" contains data for the eight Bantu languages in the Tanzanian Language Survey (questions 1 and A) in a format that is appropriate for phylogenetic parsimony analysis by the Phylip program "pars". Follow the steps outlined in section 6.3 to produce a phylogenetic tree of these languages. How does this differ from the hierarchical clustering tree? Can you identify the language named "Gwere" with any of the languages in the Ladefoged et al. dataset?