CHAPTER 17

Language and space

Dialects, maps and Multidimensional Scaling

What you will learn from this chapter:

This chapter introduces another popular method that deals with distance matrices. This method is called Multidimensional Scaling. It is a dimensionality reduction technique that represents distances between objects in a low-dimensional space. You will learn how to perform different types of metric and non-metric scaling and carry out the diagnostics of solutions by using the scree plot, the Shepard plot and goodness-of-fit measures. The chapter also shows how one can use R for creation of geographical maps with points and text labels. Finally, you will learn how to measure the correlation between two distance matrices with the help of the Mantel test. The case studies are based on geographic coordinates and several linguistic features of varieties of English all over the world.

17.1 Making maps with R

In this section you will learn how to make maps with R on the basis of geographic coordinates. To reproduce the code below, you will need the following packages:

```
> install.packages("rworldmap")
> library(Rling); library(rworldmap)
```

The dataset is called `eWAVE`. It contains a fraction of the data about varieties of English from the Electronic World Atlas of Varieties of English (Kortmann & Lunkenheimer 2013).[1] The dataset contains 20 randomly selected features from 76 varieties of English, as well as their type and region. Finally, it contains the coordinates (longitude and latitude) that reflect the approximate location of varieties.

```
> data(eWAVE)
> str(eWAVE)
'data.frame': 76 obs. of 24 variables:
```

1. URL http://ewave-atlas.org/ (last access 21.11.2014).

```
$ F1: Factor w/ 6 levels "?","A","B","C",..: 3 1 5 4 3 5 3 2
5 6 …
$ F7: Factor w/ 6 levels "?","A","B","C",..: 2 1 2 3 2 3 2 2
2 6 …
[output omitted]

$ Region: Factor w/ 8 levels "Africa","America",..: 3 6 2 3 3 5 5
5 5 6 …
$ Type: Factor w/ 5 levels "English-based Creoles",..: 3 4 5 3 3 1
3 1 1 1 …
$ Lon: num 120 177.6 -79.5 149.1 147.3 …
$ Lat: num -25.2 -17.6 39.5 -35.2 -42.9 …
```

The row names contain the names of the varieties. For example, the first ten varieties are as follows:

```
> rownames(eWAVE)[1:10]
[1] "Aboriginal English"   "Acrolectal Fiji English"
[3] "Appalachian English"  "Australian English"
[5] "Australian Vernacular English" "Bahamian Creole"
[7] "Bahamian English" "Barbadian Creole (Bajan)"
[9] "Belizean Creole" "Bislama"
```

For the present case study, only the three last variables will be relevant. First, let us use the coordinates to create a world map where the geographic locations of varieties are displayed as symbols which represent the linguistic type of varieties. We will first create a numeric vector that represents the types as numbers.

```
> code <- as.numeric(eWAVE$Type)
> code
[1] 3 4 5 3 3 1 3 1 1 1 4 1 2 4 2 4 3 4 3 3 3 5 1
[24] 5 5 5 3 4 2 1 1 1 4 4 4 3 1 4 4 1 3 4 3 3 5 3
[47] 4 2 2 5 5 4 1 3 4 1 3 1 1 5 5 1 4 3 4 2 1 1 3
[70] 4 3 2 1 3 3 3
```

Now we are ready to create our first map with the help of some functions from the package `rworldmap`.

```
> dialmap <- getMap()
> plot(dialmap)
> points(eWAVE[, 23:24], pch = code + 20, col = code, bg = code)
```

The first two lines of code are needed to create an empty map of the world. The third line adds points to the map. The contour and filling colour of the symbols correspond to the five linguistic types. We add 20 to the `pch` argument because R symbols that can be filled with different colours are in the range from 21 to 25. Finally, we provide a legend:

```
> legend("bottomleft", legend = levels(eWAVE$Type), pch = 21:25,
col = 1:5, pt.bg = 1:5)
```

The result is shown in Figure 17.1.

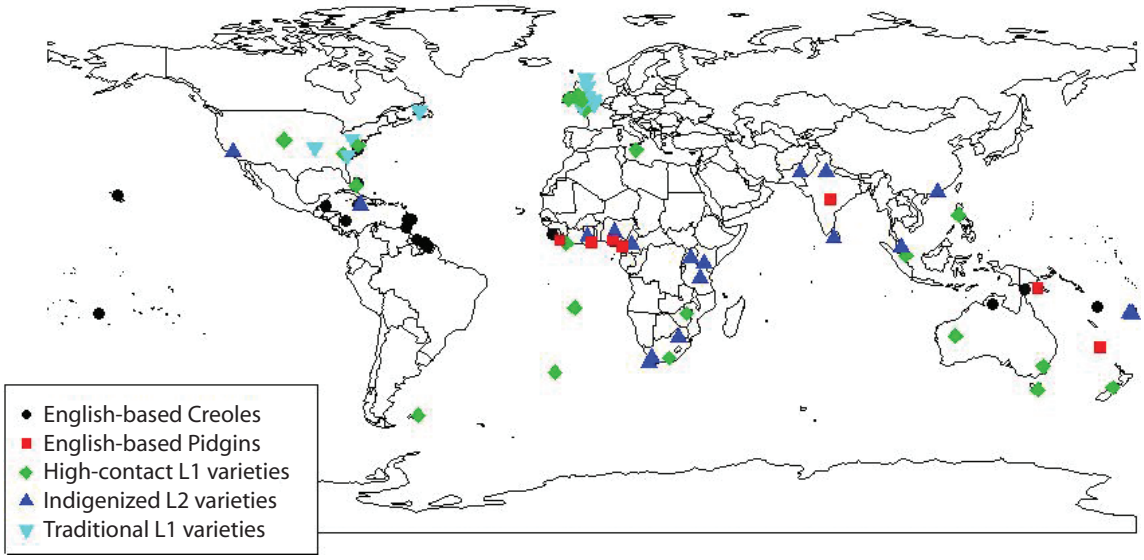

Figure 17.1. A map of the world with 76 varieties of English from eWAVE

To zoom in on a specific region, one can use `xlim()` and `ylim()`. The x coordinates specify the longitude, whereas the y coordinates stand for the latitude. As an illustration, let us zoom in on the USA mainland territory and display two varieties: Appalachian English and Chicano English. Their coordinates in `eWAVE` are found in rows 3 and 18, respectively. To zoom in on the USA territory, you can use the following code, which will create an empty map with country boundaries:

```
> plot(dialmap, xlim = c(-140, -40), ylim = c(30, 50), asp = 1)
```

You can try out different boundaries with the help of `xlim()` and `ylim()`, as well as projection types (`asp`). Next, let us plot the locations as points and add text labels, as shown in Figure 17.2:

```
> points(eWAVE[c(3, 18), 23:24], col = code[c(3, 18)], cex = 1.2,
pch = code[c(3, 18)] + 20, bg = code[c(3, 18)])
> text(eWAVE[c(3, 18), 23:24], labels = rownames(eWAVE)[c(3, 18)],
cex = 0.8, adj = c(0.5, 1.2))
```

The `adj` argument provides a small horizontal and vertical adjustment for the text labels, to avoid overplotting.

In the remaining part of the chapter, you will learn how to represent geographic and linguistic distances with the help of Multidimensional Scaling, as well as how to compute correlations between distance matrices.

Figure 17.2. Zooming in on the USA mainland territory and two U.S. varieties

More on maps in R

In this section, we have only discussed some basic plotting tools for creation of maps. There are many useful packages in R that can help you create attractive and informative maps. One of them is `ggmap`, which downloads data from Google Maps and uses a structure similar to `ggplot2`, adding layers incrementally. A relatively easy way of highlighting countries of interest in different colours by using their names or ISO codes is provided in the package `rworldmap`. See also pp. 309 – 321 in R Graphics Cookbook (Chang 2012) about how to make maps with `ggplot2`, use different colours to represent values of variables and create maps from a shapefile.

17.2 What is Multidimensional Scaling?

Multidimensional Scaling (MDS) is a method that represents distances between objects in a low-dimensional space. Imagine a table of distances between different cities. With the help of MDS, it is possible to obtain a two-dimensional representation of these distances,

similar to a geographical map, with the cities represented as points. One can also visualize non-spatial relationships, such as similarities and dissimilarities between languages, language varieties, words or phonemes. The aim of MDS is to find a low-dimensional space that would represent these objects as points in the space, such that the distances between these points would match the original (dis)similarities between the objects as precisely as possible. The larger the dissimilarity (and the smaller the similarity) between two objects, the farther apart these objects should be in the MDS space.

There are many types of MDS. The most important distinction is that between metric and non-metric MDS. The purpose of **metric** MDS is to represent the objects in a low-dimensional space, so that the original distances or dissimilarities between them are represented as precisely as possible by the distances between the points on the MDS map. Perhaps the best-known type of metric MDS is the so-called **classical MDS**, or Principal Coordinates Analysis. In **non-metric** MDS, it is the ranking of dissimilarities between the objects that matters, rather than their actual numerical values. The purpose of non-metric MDS is to find a configuration of objects such that the rankings of the MDS distances between the objects correspond as closely as possible to the rank order of the original dissimilarities between the objects. Non-metric MDS is widely used in psychometrics, when one deals with similarity and dissimilarity ratings on the ordinal scale. Less strict than metric MDS, it should be used when the ordering is more important than the actual distances. Note, however, that non-metric MDS algorithms tend to be slower than metric, so the latter may be preferred if you have a very large dataset.

The better the original distances or dissimilarities are represented in a dimensionality-reduction solution, the less information is lost. For example, a two-dimensional map may not perfectly represent the driving distances between different cities because of the Earth curvature or differences in elevation. Such loss of information is called **stress**. The amount of stress in a solution is the most important goodness-of-fit measure in MDS. The smaller the stress, the better the fit.

A solution with zero stress, however, is not necessarily the optimal one. In principle, it is possible to represent the distances between n objects with 100% accuracy in a model with $n - 1$ dimensions. For example, the distance between two points can be represented by a one-dimensional straight line, whereas all distances between three points can be perfectly represented on a two-dimensional plane, and so on. Still, the heuristic value of MDS is determined by its ability to find a small number of interpretable dimensions that account for most variation in the data. Similar to other multivariate methods, the art of MDS consists in striking the balance between parsimony and accuracy. The evaluation and diagnostics of MDS solutions will be explained below. The next section introduces the classical MDS, which will represent geographical distances on two- and three-dimensional maps, whereas Section 17.4 shows how to fit the Kruskal non-metric MDS of linguistic distances based on categorical data.

17.3 Computation and representation of geographical distances

This section shows how one can represent geographical distances with the help of classical MDS. Although this task is not particularly common in linguistic studies, it offers a simple example that illustrates the basic principles of MDS. To reproduce the code in this section, you will need the following packages:

```
> install.packages(c("Rling", "fields", "rgl", "MASS"))
> library(Rling); library(fields); library(rgl); library(MASS)
```

The dataset of English varieties `eWAVE` was introduced in the previous section. Again, only the last two columns with the longitude and latitude will be relevant.

```
> data(eWAVE)
```

To demonstrate how geographical distances can be represented by MDS, we will compute the distances from the coordinates with the help of `rdist.earth()` from the package `fields`. This function computes distances between all pairs of objects on the basis of their longitude and latitude coordinates, taking into account the Earth curvature. The result is a matrix with geographic distances. One can compute the distances in miles (the default) or kilometres (as shown below). Next, we transform the matrix into a distance matrix object.

```
> geo.dist <- rdist.earth(eWAVE[, 23:24], miles = FALSE)
> geo.dist <- as.dist(geo.dist)
```

Now we can perform MDS. The main function for classical MDS is `cmdscale()`. The first argument is the distance matrix. The argument `eig` is optional. It 'tells' R to compute the **eigenvalues** of every dimension, i.e. the contributions of all possible dimensions to explaining variance. This information is useful for deciding on the optimal number of dimensions. The default number of dimensions is two.

```
> geo.mds <- cmdscale(geo.dist, eig = TRUE) #equivalent to
cmdscale(geo.dist, k = 2, eig = TRUE)
```

The object `geo.mds` contains, among other things, coordinates of all varieties in two dimensions (the default), as well as the eigenvalues. To decide on the optimal number of dimensions, it is useful to examine a scree plot of the eigenvalues. The scree plot in Figure 17.3 shows the eigenvalues of the first 20 dimensions.

```
> barplot(geo.mds$eig[1:20], xlab = "Number of dimensions", ylab =
"Eigenvalues", main = "Scree plot")
```

The scree plot demonstrates that there is a substantial decrease in the explanatory power of new dimensions after the thirst three ones. It seems that a three-dimensional solution is the optimal one (not surprisingly). However, let us begin our visual inspection with a

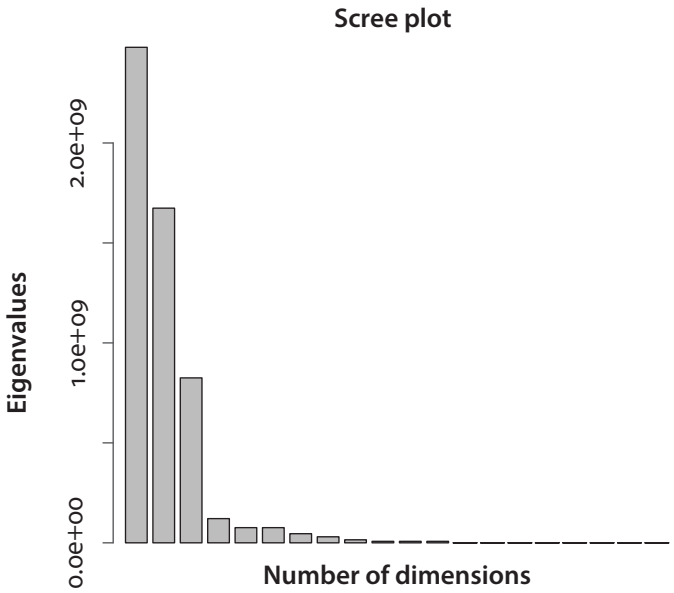

Figure 17.3. The scree plot of proportions of variance explained by the first 20 dimensions

two-dimensional plot. The code below demonstrates how one can create an empty MDS map and plot the text labels of the varieties onto it. Figure 17.4 displays the result.

```
> plot(geo.mds$points, type = "n", main = "MDS of geographic
distances between varieties of English")
> text(geo.mds$points, labels = rownames(eWAVE), cex = 0.6)
```

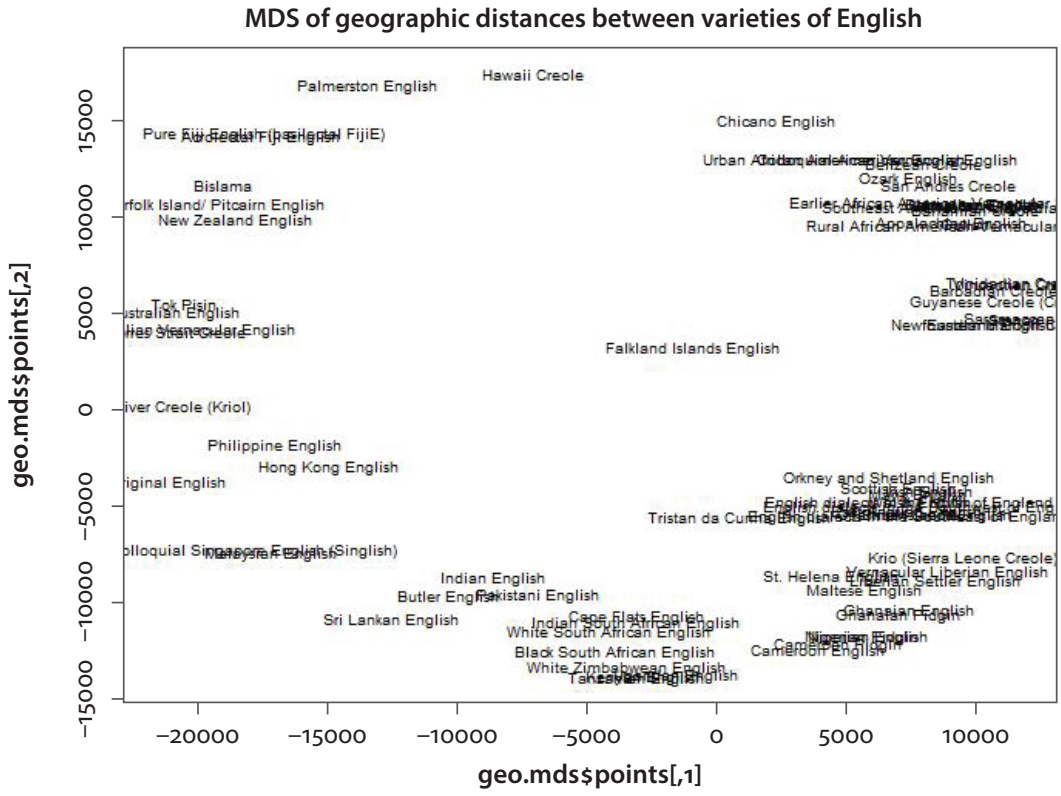

Figure 17.4. An MDS plot with text labels of the varieties of English

The plot has an empty centre and a densely populated periphery. A closer inspection reveals that the dialects are arranged in a specific order, as if one was looking on the globe from 'below', with the centre approximately at the South Pole. The upper part of the map with Americas roughly corresponds to the Western hemisphere, and the lower part with Europe, Asia and Africa to the Eastern hemisphere. The map does not represent the north/south distinctions (latitudes). To represent this information, it is necessary to add the third dimension. One can do it on a three-dimensional plot with the help of the package `rgl`. Before creating the plot, one should re-run `cmdscale()` with `k = 3` to obtain the coordinates on the third dimension.

```
> geo.mds.3d <- cmdscale(geo.dist, k = 3, eig = TRUE)
> plot3d(geo.mds.3d$points, type = "n")
> text3d(geo.mds.3d$points, texts = rownames(eWAVE), cex = 0.6)
```

The plot, shown in Figure 17.5, can be rotated with the help of the left button of the mouse or the touchpad. You can also zoom out and in by holding the right button and using the scroll wheel of the mouse or making sweeping movements with your touchpad.

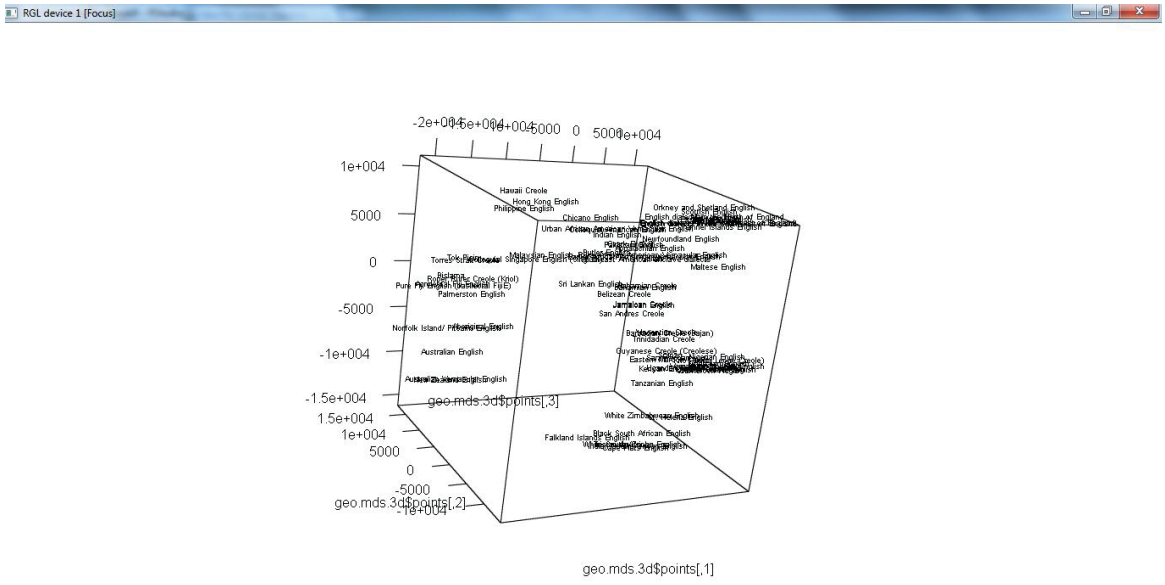

Figure 17.5. A 3D MDS plot of varieties of English based on geographic distances

A closer inspection of the 3D plot demonstrates that the varieties are plotted in a way that resembles their locations on the Earth surface. The labels, however, are located 'inside out', that is, the points that are on the left from the other points are located in 'real life' to the East from the other points. This mismatch is not a problem. It often happens in MDS that the coordinates are 'flipped', but a 'flipped' solution is just as good as an 'unflipped' one.

Although the solution looks interpretable, it is necessary to find out whether it represents the data well. The GOF component of the MDS object gives two goodness-of-fit measures, which give an idea about the quality of the model, similar to the R^2 statistic in linear regression. The higher the scores, the better.

```
> geo.mds.3d$GOF
[1] 0.7423278 0.9207068
```

Another way of estimating the goodness of fit is Kruskal's (1964) formula for computation of stress:

$$Stress = \left[\frac{\sum_{i<j} \left(d_{ij} - \hat{d}_{ij} \right)^2}{\sum_{i<j} d_{ij}^{\ 2}} \right]^{1/2}$$

where d_{ij} is the actual distance between objects *i* and j, and $\hat{d}_{ij}$ is the fitted distance between the same objects in the MDS solution. In R, this can be computed as follows:

```
> sqrt(sum((geo.dist - dist(geo.mds.3d$points))^2)/sum(geo.dist^2))
[1] 0.1432637
```

Stress indicates the opposite of the GOF measures mentioned above. The smaller the stress, the better the fit. Below are some rules of thumb for interpretation of stress values in MDS solutions:

0.2 ≤ Stress	Poor
0.1 ≤ Stress < 0.2	Fair
0.05 ≤ Stress < 0.01	Good
Stress < 0.05	Excellent

The GOF values and stress only show the general picture, but they do not tell us which individual distances are fitted well and which ones poorly. For that purpose, one can use the Shepard diagnostic plot in the `MASS` package. It shows how well the distances in the initial matrix are approximated by the fitted distances in the resulting solution. The method allows one to identify degenerate solutions and other problems (see Borg & Groenen 1997: Ch. 13).

```
> geo.sh <- Shepard(geo.dist, geo.mds.3d$points)
> plot(geo.sh, main = "Shepard plot", pch = ".")
> lines(geo.sh$x, geo.sh$yf, type = "S")
```

Figure 17.6 shows the Shepard plot of the data. The *x*-axis corresponds to the geographic distances between each pair of varieties, whereas the *y*-axis shows the distances between the varieties as they are represented by the MDS solution. The points that are remote from

Figure 17.6. The Shepard plot for diagnostics of MDS solution

the diagonal are outliers. They contribute to the stress value the most. In this example, we do not observe serious problems with the fit of individual distances.

17.4 Computation and representation of linguistic distances: The Kruskal non-metric MDS

17.4.1 Recoding the dataset

This section will demonstrate how one can compute distances based on ordinal and other types of data and represent them in non-metric MDS. To reproduce the code, you will need the following packages:

```
> install.packages(c("Rling", "cluster", "MASS", "ggplot2"))
> library(Rling); library(cluster); library(MASS); library(ggplot2)
```

We will continue our analysis of the dataset `eWAVE`:

```
> data(eWAVE)
```

The first twenty columns in eWAVE dataset represent twenty randomly selected linguistic features from the Electronic World Atlas of Varieties of English.[2] The features represent different aspects of language structure, including morphology, syntax and organization of discourse. The Atlas has its own system of coding the presence or absence of linguistic

2. See the full list of features at http://ewave-atlas.org/parameters (last access 24.06.2014).

features in varieties: 'A' means that the feature is pervasive or obligatory; 'B' says that the feature is neither pervasive nor extremely rare; 'C' means that the feature exists, but it is extremely rare; 'D' stands for an attested absence of the feature. There are two types of missing values: 'X' shows that the structural feature is not applicable given the structural make-up of the variety, and '?' indicates missing information.

This type of representation is typical of ordinal variables (see Chapter 1). It is not clear whether the difference between the categories 'A' and 'B' is the same as the difference between 'B' and 'C' or 'C' and 'D'. Ordinal categorical variables can be represented by ordered factors in R. In addition, it is necessary to recode 'X' and '?' into 'NA', so that the algorithms that will be used below can identify the missing values. This can be done as follows. First, we create a copy of the dataset. Next, the missing values are substituted with 'NA':

```
> eWAVE1 <- eWAVE
> eWAVE1[eWAVE1 == "?"] <- NA
> eWAVE1[eWAVE1 == "X"] <- NA
```

The next step is to order the factors, so that 'D' represents the smallest value, and 'A' represents the greatest value. To apply the transformation to all features simultaneously, one can use the function `lapply()`. Since the function returns a list of factors, it is necessary to combine them in a new data frame.

```
> eWAVE2 <- lapply(eWAVE1[, 1:20], function(x) ordered(x, levels =
c("D", "C", "B", "A")))
> eWAVE2 <- data.frame(eWAVE2)
```

The resulting data frame looks as follows:

```
> str(eWAVE2)
'data.frame': 76 obs. of 20 variables:
$ F1: Ord.factor w/ 4 levels "D"<"C"<"B"<"A": 3 NA 1 2 3 1 3 4…
$ F7: Ord.factor w/ 4 levels "D"<"C"<"B"<"A": 4 NA 4 3 4 3 4 4…
$ F48: Ord.factor w/ 4 levels "D"<"C"<"B"<"A": 4 3 3 NA NA 2 3 …
[output omitted]
```

17.4.2 Computation of Gower distances

In Chapter 15, the distances were computed between vectors with continuous values. Here, the data are ordinal. To compute the distances between the varieties, we will use the Gower general coefficient of similarity, which can be computed for all kinds of variables, including categorical ones. In its most general form, it looks as follows (Gower 1971: 861):

$$S_{ij} = \sum_{k=1}^{v} s_{ijk} w_k \,/\, \sum_{k=1}^{v} \delta_{ijk} w_k$$

where S_{ij} is the general similarity between observations *i* and *j*, *v* is the number of variables, s_{ijk} is the similarity between *i* and *j* with regard to the variable *k*, and δ_{ijk} is the possibility of making comparison between *i* and *j* regarding *k*. This value is equal to 1 if both exemplars have non-missing values, and 0 otherwise. Finally, w_k is the weight of the variable *k*, which can be specified by the researcher. By default, all weights are equal to 1.

For ordinal and numeric variables, s_{ijk} is defined as 1 minus the absolute difference of both values divided by the total range of that variable. The values of ordinal variables are replaced by their integer representations. For example, in the English varieties data, 'A' will correspond to 4 and 'D' will correspond to 1.

For nominal variables, including binary ones, s_{ijk} equals 1 if the values of *i* and *j* are identical, and 0 if the values are different. From this follows that the Gower similarity coefficient returns the proportion of overlapping values in two vectors with categorical values with regard to the total number of possible comparisons.[3]

The Gower coefficient shows the degree of similarity. However, for MDS we need the opposite of similarities, namely, dissimilarities that can be represented as distances. The distances between exemplars are calculated by subtracting the similarity scores from 1.

The distance metric based on the Gower similarity coefficient, which is referred to as the Gower distance, is implemented in the `daisy()` function in the `cluster` package. This metric is chosen automatically if at least one variable in the data is not numerical.

```
> ling.dist <- daisy(eWAVE2) # equivalent to daisy(eWAVE2, metric
= "gower")
> summary(ling.dist)
2850 dissimilarities, summarized:
Min.    1st Qu. Median  Mean    3rd Qu. Max.    NA's
0.00000 0.28070 0.35556 0.37555 0.45000 1.00000 8
Metric: mixed ; Types = O, O, O, O, O, O, O, O, O, O, O, O, O, O,
O, O, O, O, O, O
Number of objects: 76
```

There is one small problem with the distances. The MDS algorithms that will be introduced below do not accept missing values. As can be seen from the summary, the distance matrix contains eight 'NA's. They can be replaced with the mean distance value. Another solution would be to inspect the data and remove the varieties that contain too many missing values.

```
> ling.dist1 <- ling.dist
> ling.dist1[is.na(ling.dist)] <- mean(ling.dist, na.rm = TRUE)
```

3. In addition, a binary variable with values '1' and '0' can be treated asymmetrically. In that case, the weighting sigma equals 0 if both observations have '0'. See the help page of the function `daisy()` in the package `cluster` for more information and examples.

In addition, the MDS algorithm that we are going to use does not accept zero or negative distances. The zero values can be replaced with very small positive numbers:

```
> ling.dist2 <-ling.dist1
> ling.dist2[ling.dist2 == 0] <- 0.001
```

We will discuss a popular non-metric approach developed by Kruskal (1964). You will need the function `isoMDS()` from the package `MASS`. Its main arguments are a distance matrix and the preferred number of dimensions. Let us begin with a two-dimensional solution (the default). Since this function does not deal with zero distances, the distance matrix `ling.dist2` will be used:

```
> ling.mds.kr <- isoMDS(ling.dist2) #equivalent to isoMDS(ling.
dist2, k = 2)
initial value 32.470515
iter 5 value 24.586167
final value 24.416368
converged
```

Number of iterations

Sometimes, the algorithm may stop after 50 iterations, as in the example with imaginary data below:

```
> test <- isoMDS(test.dist, k = 8) # do not run
initial value 6.601167
iter 5 value 1.934208
…
iter 50 value 0.428581
final value 0.428581
stopped after 50 iterations
```

The algorithm tries to find a stress-optimal configuration by re-scaling the data again and again until the next possible improvement is below a convergence criterion. The default number of iterations is 50. In this example, this number was not enough for the algorithm to reach the point of no further substantial improvement. To increase the number of iterations from 50 to 100 or another number, one can use the following code:

```
> test <- isoMDS(test.dist, k = 8, maxit = 100)
```

(*Continued*)

```
initial value 6.601167
iter  5 value 1.934208
…
iter 60 value 0.394103
final value 0.388977
converged
```

The solution can be represented with the help of `qplot()` in `ggplot2`. This function has been chosen because it can easily represent the text labels in different colours depending on the type of the variety, as shown below, or any other variable. It will also automatically add a legend. The resulting plot is shown in Figure 17.7.

```
> qplot(ling.mds.kr$points[, 1], ling.mds.kr$points[, 2], label =
rownames(eWAVE), cex = 0.5, col = eWAVE$Type, geom = "text")
```

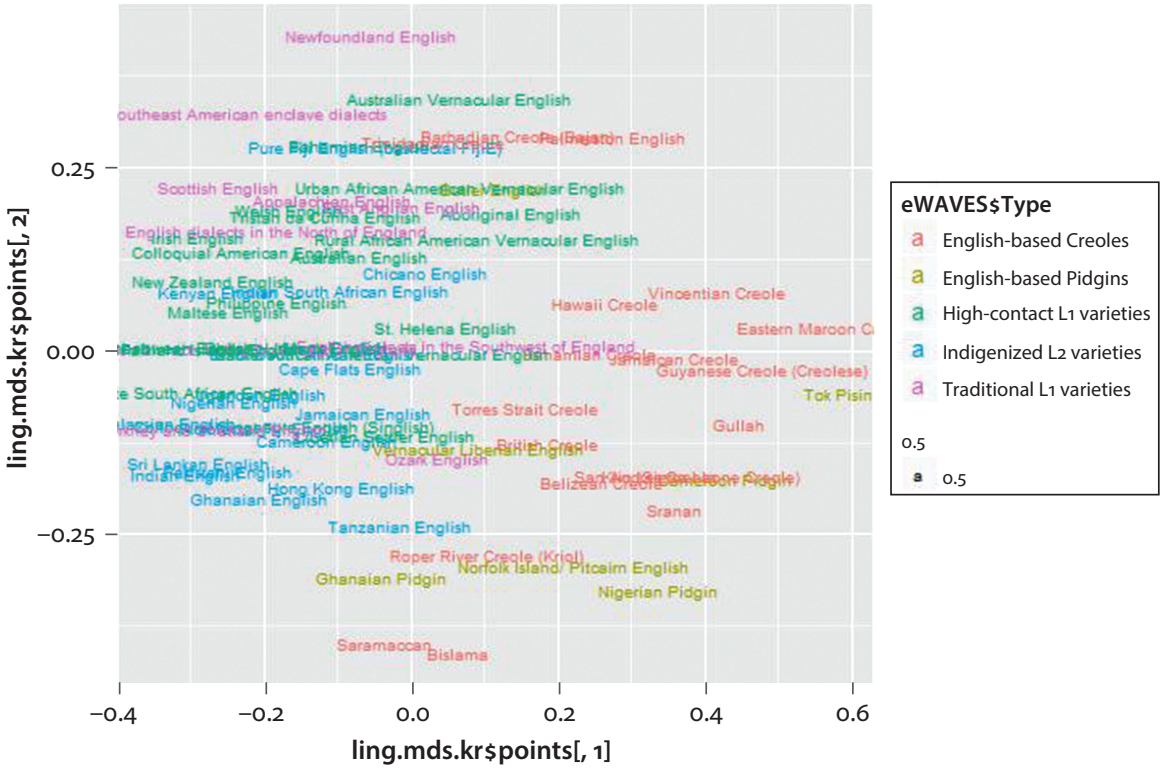

Figure 17.7. A two-dimensional Kruskal non-metric MDS of English varieties based on twenty linguistic features

The plot demonstrates a continuum from the traditional L1 varieties (mostly in the upper left sector) via the high-contact L1 varieties and indigenized L2 varieties to the creoles and pidgins. Although the corresponding areas overlap, the general tendency is still observed.

The next question is whether the two-dimensional representation is optimal. We can test it by fitting MDS models for different numbers of dimensions and comparing the relative decrease in the stress value with the help of a scree plot. The stress of the solution is printed out as the 'final value'. The stress of the two-dimensional solution was 24.42. This value is large. This means one loses much information. In this situation, one can increase the number of dimensions until acceptable quality is achieved.

Warning: degenerate MDS solutions

In some non-metric MDS solutions, the stress may be very low, but the solution may be poor. Consider the Shepard plot in Figure 17.8. The MDS stress is almost 0, but the solution is degenerate because the algorithm dichotomizes all fitted distances into 0 and 20. See Borg & Groenen (1997: 270–273) for a detailed explanation. A graphical indication of such problems is the presence of long horizontal lines.

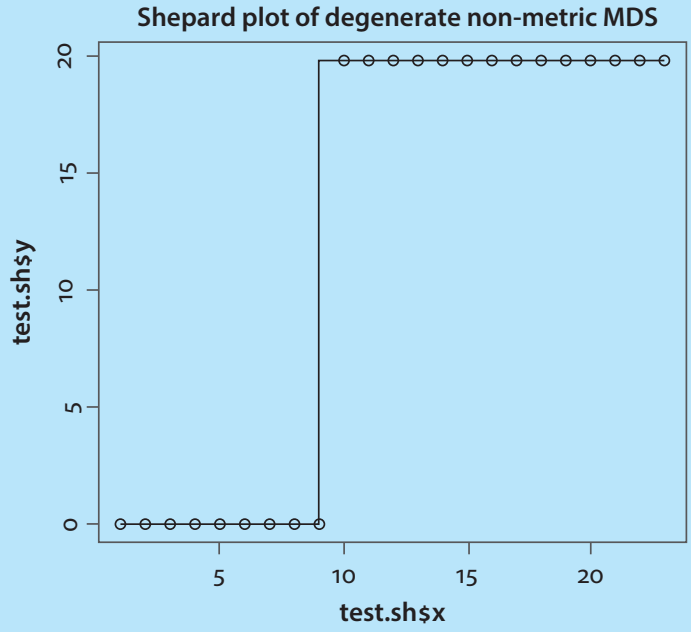

Figure 17.8. The Shepard plot of a degenerate MDS solution

A general recommendation in such situations is to use a stricter approach, that is, metric MDS.

Ties in non-metric MDS: how to treat similar distances?

When a distance matrix contains identical values, one can speak of **ties**. There are two main ways of tie treatment in MDS: primary (weak) and secondary (strong). If the primary treatment is used, ties impose no restrictions on the distances in the resulting MDS solution. The fact that the input distances are identical is simply not taken into account. The secondary treatment tries to represent equal input distances as equal distances in the resulting MDS model. To decide on the treatment method, one should consider how the identical distances arise in the first place and whether this is a true property of the objects or a mere artifact of measurement (Borg & Groenen 1997: 212–213). In case you want to take the identical values into account (for instance, if you have very few different values in your distance metric, and cannot afford losing any information), you can use the secondary approach to ties. This option is not available in `isoMDS()`, but you can use the `smacofSym()` function in the package `smacof`. The code will be as follows: `smacofSym(yourDistMatrix, type = "ordinal", ties = "secondary")`.

17.5 The Mantel test for distance matrices

An interesting question is whether one can find any correlation between the geographic distances and the linguistic differences between the varieties. One can compute simple correlations between the values in two distance matrices. However, since the distances are between pairs and the observations are not independent, it is not advisable to use a standard correlation test. Instead, one should use the Mantel test. Here we will discuss a version of this test based on permutation from the package `vegan`. To perform the test, you will need the distance objects from the previous case studies (Sections 17.3 and 17.4).

```
> install.packages("vegan")
> library(vegan)
```

```
> mantel(geo.dist, ling.dist, na.rm = TRUE) #equivalent to mantel(geo.dist, ling.dist, method = "pearson", permutations = 999, na.rm = TRUE)
```

```
Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = geo.dist, ydis = ling.dist, na.rm = TRUE)

Mantel statistic r: 0.1213
      Significance: 0.001

Upper quantiles of permutations (null model):
90%     95%     97.5%   99%
0.0351  0.0525  0.0679  0.0877

Based on 999 permutations
```

Note that one has to add `na.rm = TRUE` because the original linguistic distance matrix `ling.dist` contains missing values. The default statistic is the Mantel statistic, which is in fact identical to the Pearson correlation coefficient *r*. To compute rank-based correlation coefficients, type in `method = "spearman"` or `method = "kendall"`. In all cases, the correlation is significant and positive, but weak. This means that there are other factors that influence the linguistic characteristics of the varieties. One of them is the amount and type of language contact.

17.6 Summary

Multidimensional Scaling is a convenient tool for visual exploration of multivariate data. Its advantage in comparison with cluster analysis is that MDS can represent both groupings and interpretable dimensions of variation, whereas cluster analysis only returns groupings. This is especially useful when one wants to model a continuum of linguistic categories or varieties.

How to present MDS results

When reporting the results of MDS, one should not forget to mention the quality of the solution (stress), the type of distance metric and the method. Of course, all relevant MDS maps should be provided and interpreted.

One of the weaknesses of traditional MDS, however, is that there is no straightforward way to interpret the location of the points with regard to the input variables. In the next two chapters, we will discuss several multivariate methods that allow one to retain information about the input variables and combine them with the information about the groups of objects. These methods are Principal Components Analysis, Factor Analysis (for quantitative data) and Correspondence Analysis (for categorical data).

More flavours of MDS

The `smacof` package also contains many useful functions for individual differences scaling, constrained MDS and other methods. For more theoretical information about those and other types of MDS, see Borg & Groenen (1997) and Cox & Cox (2001).