# Chapter 4
# Cluster Analysis


## 4.1 Objectives

In most cases, data exploration (Chap. 2) and the computation of association matrices (Chap. 3) are preliminary steps towards deeper analyses. In this chapter you will go further by experimenting one of the large groups of analytical methods used in ecology: clustering. Practically, you will:

- learn how to choose among various clustering methods and compute them;
- apply these techniques to the Doubs River data to identify groups of sites and fish species.
- explore two methods of constrained clustering, a powerful modelling approach where the clustering process is constrained by an external data set.

## 4.2 Clustering Overview

The objective of clustering is to recognize discontinuous subsets in an environment that is sometimes discrete (as in taxonomy), but most often perceived as continuous in ecology. This requires some degree of abstraction, but ecologists may want to get a simplified, structured view of their data; generating a typology is one way to achieve that goal. In some instances, typologies are compared to independent classifications (based on theory or on other typologies obtained from independent data). What we present here is a collection of methods used to decide whether objects are similar enough to be allocated to a group, and identify the distinctions or separations between groups.

*Clustering* consists in partitioning the collection of objects (or descriptors in R-mode) under study. A hard *partition* is a division of a set (collection) into subsets, such that each object or descriptor belongs to one and only one subset for that partition (Legendre and Rogers 1972). For instance, a species cannot be

© Springer International Publishing AG, part of Springer Nature 2018

D. Borcard et al., *Numerical Ecology with R*, Use R!,
https://doi.org/10.1007/978-3-319-71404-2_4

simultaneously the member of two genera: membership is binary (0 or 1). Some methods, less commonly used, consider fuzzy partitions, in which membership is continuous (between 0 and 1). Depending on the clustering model, the result can be a single partition or a series of hierarchically nested partitions. Except for its constrained version, clustering is *not* a statistical method in the strict sense since it does not test any a priori hypothesis. Post-hoc cluster robustness tests are available, however (see Sect. 4.7.3.3). Clustering helps bring out some features hidden in the data; it is the user who decides if these structures are interesting and worth interpreting in ecological terms.

Note that most clustering methods are computed from association matrices, which stresses the importance of the choice of an appropriate association coefficient.

Clustering methods differ by their algorithms, which are the effective methods for solving a problem using a finite sequence of instructions. One can recognize the following families of clustering methods:

1. *Sequential or simultaneous algorithms.* Most clustering algorithms are sequential and consist in the repetition of a given procedure until all objects have found their place. The less frequent simultaneous algorithms find the solution in a single step.
2. *Agglomerative or divisive.* Among the sequential algorithms, agglomerative procedures begin with the discontinuous collection of objects, which are successively grouped into larger and larger clusters until a single, all-encompassing cluster is obtained. Divisive methods, on the contrary, start with the collection of objects considered as one single group, and divide it into subgroups, and so on until the objects are completely separated. In either case it is left to the user to decide which of the intermediate partitions should be retained, given the problem under study.
3. *Monothetic versus polythetic*. Divisive methods may be monothetic or polythetic. Monothetic methods use a single descriptor (the one that is considered the best for that level) at each step for partitioning, whereas polythetic methods use all descriptors; in most polythetic cases, the descriptors are combined into an association matrix.
4. *Hierarchical versus non-hierarchical methods.* In hierarchical methods, the members of inferior-ranking clusters become members of larger, higher-ranking clusters. Most of the time, hierarchical methods produce non-overlapping clusters. Non-hierarchical methods (e.g. *k*-means partitioning) produce a single partition, without any hierarchy among the groups.
5. *Probabilistic versus non-probabilistic methods.* Probabilistic methods define groups in such a way that the within-group association matrices have a given probability of being homogeneous. Probabilistic methods are sometimes used to define species associations.
6. *Unconstrained or constrained methods*. Unconstrained clustering relies upon the information of a single data set, whereas constrained clustering uses two matrices: the one being clustered, and a second one containing a set of explanatory variables which provides a constraint (or guidance) as to where to group or divide the data of the first matrix.

These categories are not represented equally in the ecologist's toolbox. Most methods presented below are sequential, agglomerative and hierarchical (Sects. 4.3, 4.3.1, 4.3.2, 4.4, 4.5 and 4.6), but others, like *k*-means partitioning, are divisive and non-hierarchical (Sect. 4.8). Two methods are of special interest: Ward's hierarchical clustering and *k*-means partitioning are both least-squares methods. That characteristic relates them to the linear model. In addition, we will explore two methods of constrained clustering, one with sequential constraint (Sect. 4.14) and the other, more general, called multivariate regression tree analysis (MRT, Sect. 4.12).

Hierarchical clustering results are generally represented as dendrograms or similar tree-like graphs. Non-hierarchical procedures produce groups of objects (or variables), which may either be used in further analyses, presented as end-results (for instance species associations) or, when the project has a spatial component, mapped on the area under study.

A clustering of the sites of a species data set is informative by itself, but ecologists often want to interpret it by means of external, environmental variables. We will explore two ways of achieving this goal (Sect. 4.9).

Although most methods in this chapter will be applied to sites, clustering can also be applied to species in order to define species assemblages. Section 4.10 addresses this topic.

The search for indicator or characteristic species in groups of sites is a particularly important question in fundamental and applied ecology. It is presented in Sect. 4.11.

Finally, a brief section (Sect. 4.15) will be devoted to two methods of fuzzy clustering, a non-hierarchical approach that considers partial memberships of objects to clusters.

Before entering the subject, let us prepare our **R** session by loading the necessary packages and preparing the data tables.

```
# Load the required packages
library(ade4)
library(adespatial)
library(vegan)
library(gclus)
library(cluster)
library(pvclust)
library(RColorBrewer)
library(labdsv)
library(rioja)
library(indicspecies)
library(mvpart)
library(MVPARTwrap)
library(dendextend)
library(vegclust)
library(colorspace)
library(agricolae)
library(picante)
```

```
# Source additional functions that will be used later in this
# chapter. Our scripts assume that files to be read are in
# the working directory.
source("drawmap.R")
source("drawmap3.R")
source("hcoplot.R")
source("test.a.R")
source("coldiss.R")
source("bartlett.perm.R")
source("boxplerk.R")
source("boxplert.R")

# Function to compute a binary dissimilarity matrix from clusters
grpdist <- function(X) {
  require(cluster)
  gr <- as.data.frame(as.factor(X))
  distgr <- daisy(gr, "gower")
  distgr
}

# Load the data
# File Doubs.Rdata is assumed to be in the working directory
load("Doubs.Rdata")
# Remove empty site 8
spe <- spe[-8, ]
env <- env[-8, ]
spa <- spa[-8, ]
latlong <- latlong[-8, ]
```

## 4.3 Hierarchical Clustering Based on Links

### *4.3.1 Single Linkage Agglomerative Clustering*

Also called *nearest neighbour sorting*, this method agglomerates objects on the basis of their shortest pairwise dissimilarities (or greatest similarities): the fusion of an object (or a group) with a group at a given similarity (or dissimilarity) level only requires that one object of each of the two groups about to agglomerate be linked to one another at that level. Two groups agglomerate at the dissimilarity separating the closest pair of their members. This makes agglomeration easy. Consequently, the dendrogram resulting of a single linkage clustering often shows chaining of objects: a pair is linked to a third object, which in turn is linked with another one, and so on. The result may therefore be difficult to interpret in terms of partitions, but gradients are revealed quite clearly. The list of the first connections making an object member of a cluster, or allowing two clusters to fuse, is called the *chain of primary connections*; this chain forms the *minimum spanning tree* (MST). This entity is presented here and will be used in later analyses (Chap. 7).

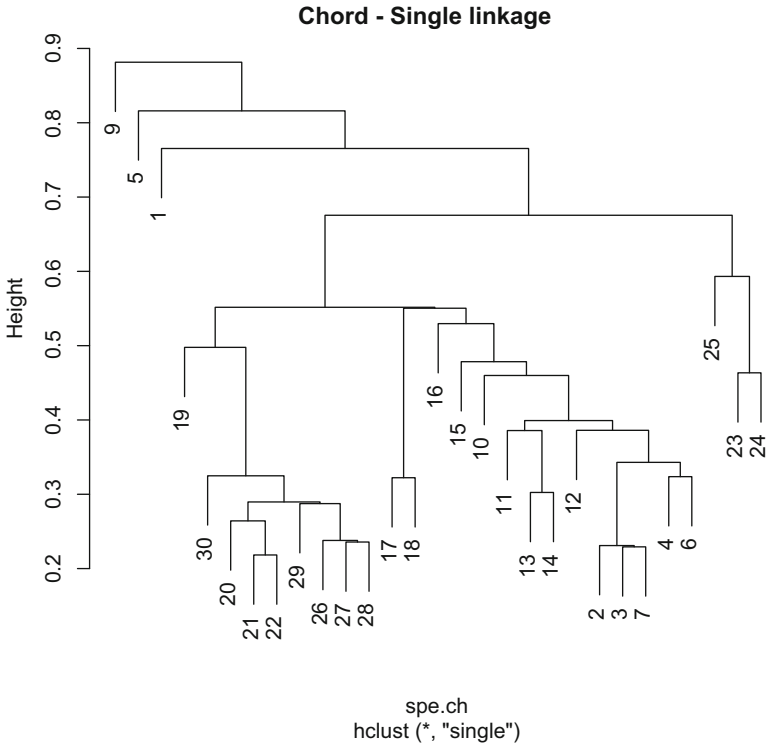


**Fig. 4.1** Single linkage agglomerative clustering of a matrix of chord distances among sites (species data)

Common hierarchical clustering methods are available through the function **`hclust()`** of the **`stats`** package. You will now compute and illustrate (Fig. 4.1) your first cluster analysis on the basis of an association matrix computed in Chap. 3 and recomputed here for convenience:

```
# Compute matrix of chord distance among sites
spe.norm <- decostand(spe, "normalize")
spe.ch <- vegdist(spe.norm, "euc")

# Attach site names to object of class 'dist'
attr(spe.ch, "labels") <- rownames(spe)

# Compute single linkage agglomerative clustering
spe.ch.single <- hclust(spe.ch, method = "single")
# Plot a dendrogram using the default options
plot(spe.ch.single,
   labels = rownames(spe),
   main = "Chord - Single linkage")
```

*Based on this first result, how would you describe the data set? Do you see a single simple gradient, or else distinguishable groups of sites? Can you identify some chaining of the sites? What about sites 1, 5 and 9?*

### 4.3.2 Complete Linkage Agglomerative Clustering

Contrary to single linkage clustering, complete linkage clustering (also called *furthest neighbour sorting*) allows an object (or a group) to agglomerate with another group only at the dissimilarity corresponding to that of the most distant pair of objects; thus, *a fortiori*, all members of the two groups are linked (Fig. 4.2):

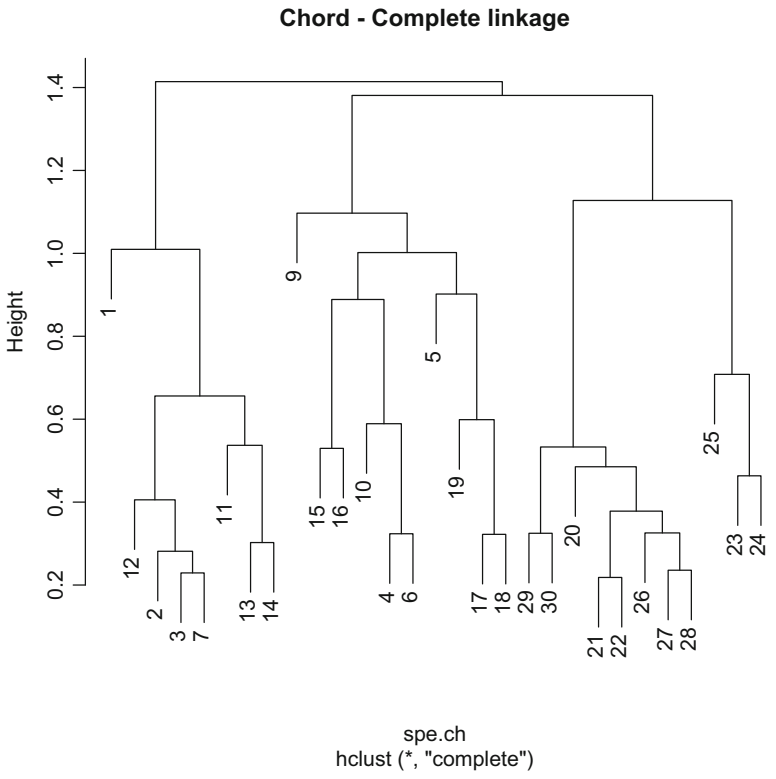


**Fig. 4.2** Complete linkage agglomerative clustering of a matrix of chord distance among sites (species data)

```
# Compute complete-linkage agglomerative clustering
spe.ch.complete <- hclust(spe.ch, method = "complete")
plot(spe.ch.complete,
   labels = rownames(spe),
   main = "Chord - Complete linkage")
```

> *Given that these are sites along a river (with the numbers following the stream), does this result tend to place sites that are neighbours along the river in the same groups?*
> *How can two perfectly valid clustering methods produce such different results when applied to the same data?*

The comparison between the two dendrograms (Figs. 4.1 and 4.2) shows the difference in the philosophy and the results of the two methods: single linkage allows an object to agglomerate easily to a group, since a link to a single object of the group suffices to induce fusion. This is a "closest friend" procedure, so to say. The resulting dendrogram does not always show clearly separated groups, but can be used to identify gradients in the data. At the opposite, complete linkage clustering is much more contrasting. A group admits a new member only at a dissimilarity corresponding to the furthest object of the group: one could say that the admission requires unanimity of the members of the group. It follows that the larger a group is, the more difficult it is to agglomerate with it. Complete linkage, therefore, tends to produce many small separate groups, which tend to be rather spherical in multivariate space and agglomerate at large distances. Therefore, this method is interesting to search for and identify discontinuities in data.

## 4.4 Average Agglomerative Clustering

This family comprises four methods that are based on average dissimilarities among objects or on centroids of clusters. The differences among them are in the way of computing the positions of the groups (arithmetic average versus centroids) and in the weighting or non-weighting of the groups according to the number of objects they contain when computing fusion levels. Table 4.1 summarizes their names and properties.

The best-known method of this family, UPGMA, allows an object to join a group at the mean of the dissimilarities between this object and all members of the group. When two groups join, they do it at the mean of the dissimilarities between all members of one group and all members of the other. Let us apply it to our data (Fig. 4.3):

**Table 4.1** The four methods of average clustering. The names in quotes are the corresponding arguments of function **hclust()**

| | Arithmetic average | Centroid clustering |
|---|---|---|
| Equal weights | Unweighted pair-group method using arithmetic averages (UPGMA) "average" | Unweighted pair-group method using centroids (UPGMC) "centroid" |
| Unequal weights | Weighted pair-group method using arithmetic averages (WPGMA) "mcquitty" | Weighted pair-group method using centroids (WPGMC) "median" |

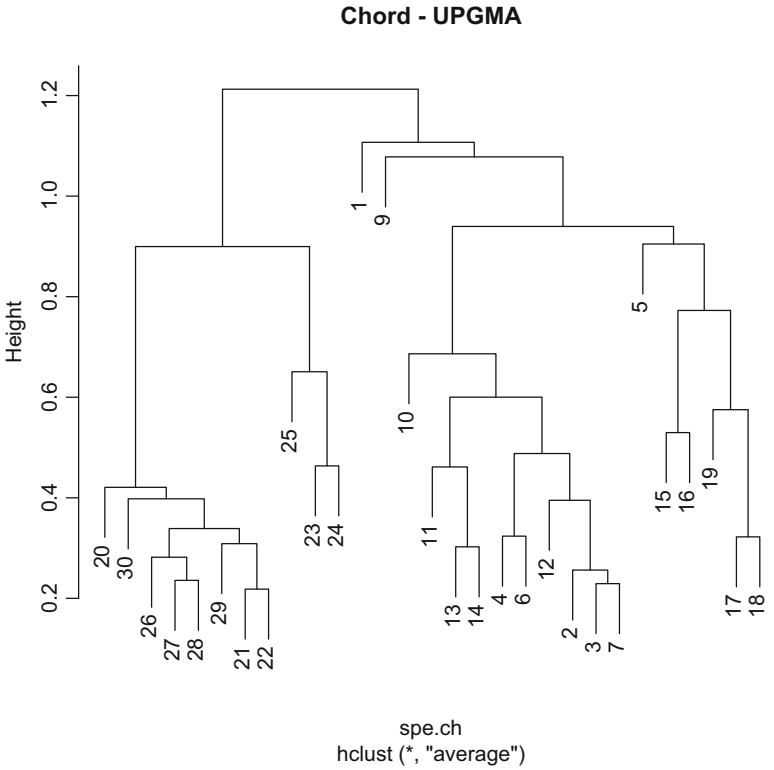


**Fig. 4.3** UPGMA clustering of a matrix of chord distance among sites (species data)

*The result looks somewhat intermediate between a single and a complete linkage clustering. This is often the case.*

```
# Compute UPGMA agglomerative clustering
spe.ch.UPGMA <- hclust(spe.ch, method = "average")
plot(spe.ch.UPGMA,
   labels = rownames(spe),
   main = "Chord - UPGMA")
```

```
# Compute centroid clustering
spe.ch.centroid <- hclust(spe.ch, method = "centroid")
plot(spe.ch.centroid,
   labels = rownames(spe),
   main = "Chord - Centroid")
```

> *The resulting dendrogram is an ecologist's nightmare. Legendre and Legendre (2012, p. 376) explain how reversals are produced and suggest interpreting them as polychotomies rather than dichotomies.*

Note that UPGMC and WPGMC can sometimes lead to reversals in the dendrograms. The result no longer forms a series of nested partitions and may be difficult to interpret. An example is obtained as follows (Fig. 4.4):

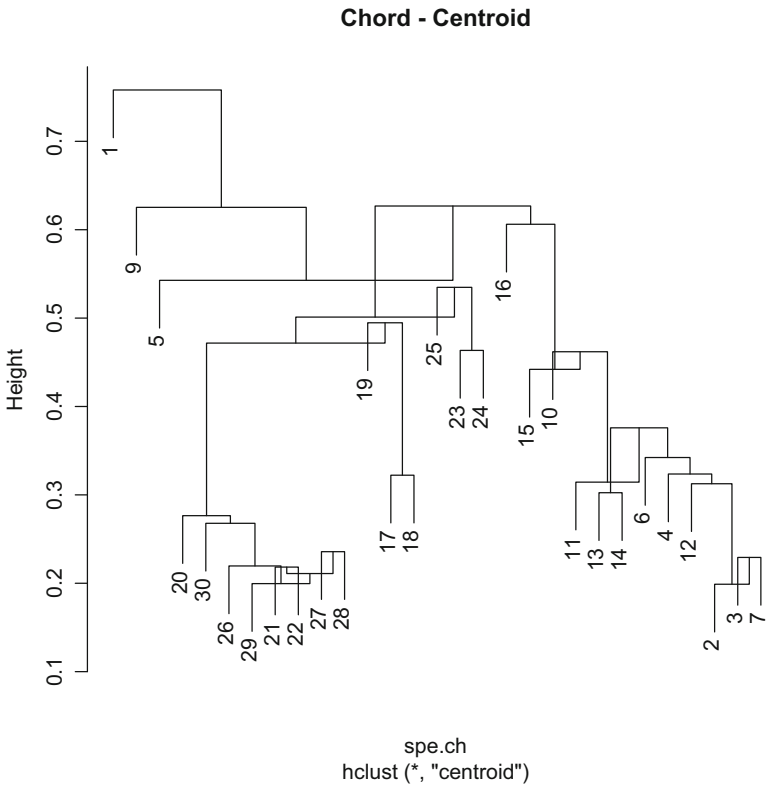


**Fig. 4.4** UPGMC clustering of a matrix of chord distance among sites (species data)

## 4.5 Ward's Minimum Variance Clustering

This method is based on the linear model criterion of least squares. The objective is to define groups in such a way that the within-group sum of squares (i.e., the squared error of ANOVA) is minimized. The within-cluster sum of squared errors can be computed as the sum of the squared distances among members of a cluster divided by the number of objects. Note also that although the computation of within-group sums-of-squares is based on a Euclidean model, the Ward method will produce meaningful results from dissimilarities that are Euclidean or not.

In the literature, two different algorithms are found for Ward clustering; one implements Ward's (1963) minimum variance clustering criterion, the other does not (Murtagh and Legendre 2014). Function **`hclust()`** was modified in R 3.1.1; `method = "ward.D2"` now implements the Ward (1963) criterion where dissimilarities are squared before cluster updating, whereas `method = "ward.D"` does not implement that criterion. The latter was implemented by **`hclust()`** with `method = "ward"` in **R** versions up to 3.0.3. Fig. 4.5 shows the dendrogram resulting from a `ward.D2` clustering.

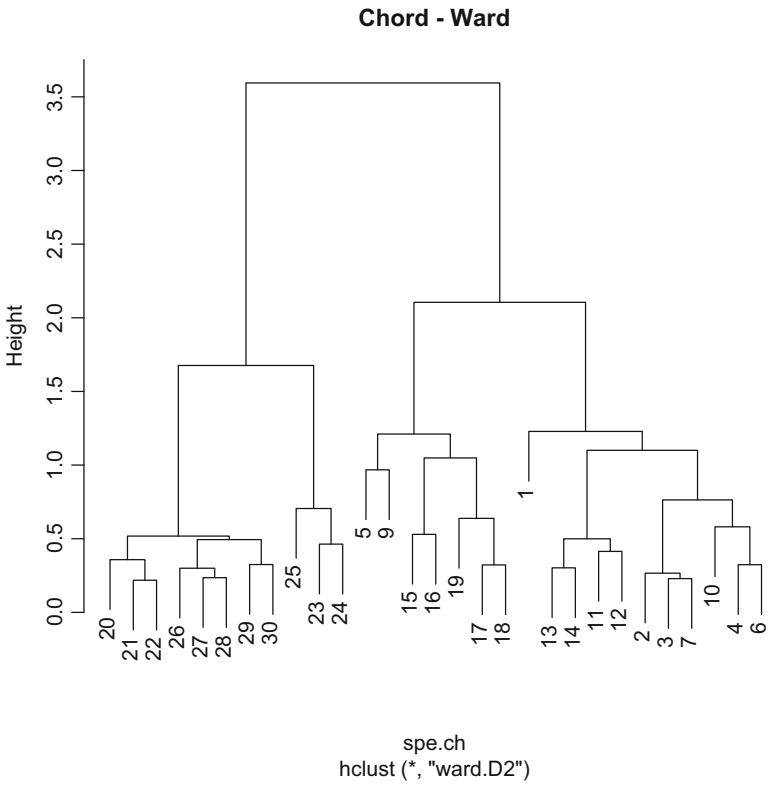


**Fig. 4.5** Ward clustering of a matrix of chord distance among sites (species data)

```
# Compute Ward's minimum variance clustering
spe.ch.ward <- hclust(spe.ch, method = "ward.D2")
plot(spe.ch.ward,
   main = "Chord - Ward")
```

*Hint* *Below you will see more options of the* **plot()** *function which produces dendrograms of objects of class* **hclust**. *Another path is to change the class of such an object using function* **as.dendrogram()**, *which opens yet more possibilities. Type* **?dendrogram** *for details.*

## 4.6 Flexible Clustering

Lance and Williams (1966, 1967) proposed a model encompassing all the clustering methods described above, which are obtained by changing the values of four parameters $\alpha_h$, $\alpha_i$, $\beta$ and $\gamma$. See Legendre and Legendre (2012, p. 370). **hclust()** is implemented using the Lance and Williams algorithm. As an alternative to the examples above, flexible clustering is available in the **R** package **cluster**, function **agnes()**, using arguments `method` and `par.method`. In **agnes()**, flexible clustering is called by argument `method = "flexible"` and the parameters are given to the argument `par.method` as a numeric vector of length 1 ($\alpha_h$ only), 3 ($\alpha_h$, $\alpha_i$ and $\beta$, with $\gamma = 0$) or 4. In the simplest application of this rather complex clustering method, flexibility is commanded by the value of parameter $\beta$, hence the name "beta-flexible clustering". Let us illustrate this by the computation of a flexible clustering with $\beta = -0.25$. If one provides only one value to `par.method`, **agnes()** considers it as the value of $\alpha_h$ in a context where $\alpha_h = \alpha_i = (1 - \beta)/2$ and $\gamma = 0$ (Legendre and Legendre 2012 p. 370). Therefore, to obtain, as in the following example, $\beta = -0.25$, the value to give is `par.method = 0.625` because $\alpha_h = (1 - \beta)/2 = (1 - (-0.25))/2 = 0.625$. See the documentation file of **agnes()** for more details.

Beta-flexible clustering with beta $= -0.25$ is computed as follows (Fig. 4.6):

```
# Compute beta-flexible clustering using cluster::agnes()
# beta = -0.25
spe.ch.beta2 <- agnes(spe.ch, method = "flexible",
  par.method = 0.625)
# Change the class of agnes object
class(spe.ch.beta2)     # [1] "agnes" "twins"
spe.ch.beta2 <- as.hclust(spe.ch.beta2)
class(spe.ch.beta1)     # [1] "hclust"
plot(spe.ch.beta2,
   labels = rownames(spe),
   main = "Chord - Beta-flexible (beta=-0.25)")
```

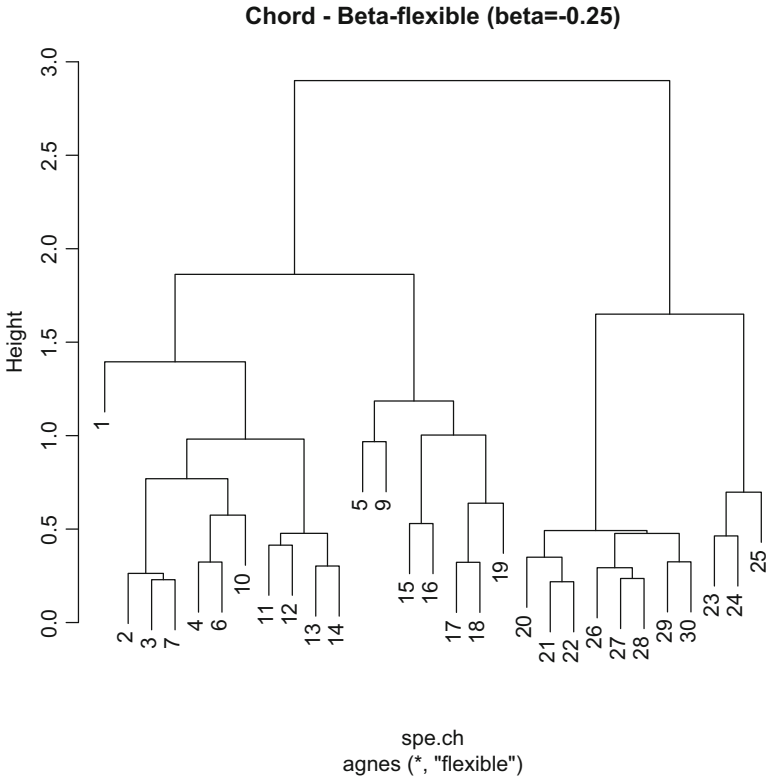


**Fig. 4.6** Beta-flexible clustering of a matrix of chord distance among sites (species data)

*Hint* *To see the influence of beta on the shape of the dendrogram, apply the complete version of the R code (online material).*

*Compare this dendrogram to the previous one (Ward method).*

## 4.7 Interpreting and Comparing Hierarchical Clustering Results

### *4.7.1 Introduction*

Remember that unconstrained clustering is a heuristic procedure, not a statistical test. The choices of an association coefficient and a clustering method influence the result. This stresses the importance of choosing a method that is consistent with the aims of the analysis. The objects produced by **`hclust()`** contain the

information necessary to fully describe the clustering results and draw the dendrogram. To display the list of items available in the output object, type **`summary(object)`** where "object" is the clustering result.

This information can also be used to help interpret and compare clustering results. We will now explore several possibilities offered by **R** for this purpose.

### *4.7.2 Cophenetic Correlation*

The cophenetic distance between two objects in a dendrogram is the distance where the two objects become members of the same group. Locate any two objects, start from one, and "climb up the tree" to the first node leading down to the second object: the level of that node along the distance scale is the cophenetic distance between the two objects. A cophenetic matrix is a matrix representing the cophenetic distances among all pairs of objects. A Pearson's *r* correlation, called the *cophenetic correlation* in this context, can be computed between the original dissimilarity matrix and the cophenetic matrix. The method with the highest cophenetic correlation may be seen as the one that produces the clustering model that retains most of the information contained in the dissimilarity matrix. This does not necessarily mean, however, that this clustering model is the most adequate for the researcher's goal.

Of course, the cophenetic correlation cannot be tested for significance, since the cophenetic matrix is derived from the original dissimilarity matrix. The two sets of distances are not independent. Furthermore, the cophenetic correlation depends strongly on the clustering method used, in addition to the data.

As an example, let us compute the cophenetic matrix and correlation of four clustering results presented above, by means of the function **`cophenetic()`** of package **`stats`**.

```
# Single linkage clustering
spe.ch.single.coph <- cophenetic(spe.ch.single)
cor(spe.ch, spe.ch.single.coph)
# Complete linkage clustering
spe.ch.comp.coph <- cophenetic(spe.ch.complete)
cor(spe.ch, spe.ch.comp.coph)
# Average clustering
spe.ch.UPGMA.coph <- cophenetic(spe.ch.UPGMA)
cor(spe.ch, spe.ch.UPGMA.coph)
# Ward clustering
spe.ch.ward.coph <- cophenetic(spe.ch.ward)
cor(spe.ch, spe.ch.ward.coph)
```

> *Which dendrogram retains the closest relationship to the chord distance matrix? Cophenetic correlations can also be computed using Spearman or Kendall correlations:*

```
cor(spe.ch, spe.ch.ward.coph, method = "spearman")
```

To illustrate the relationship between a dissimilarity matrix and a set of cophenetic matrices obtained from various methods, one can draw Shepard-like diagrams (Legendre and Legendre 2012 p. 414) by plotting the original dissimilarities against the cophenetic distances (Fig. 4.7):

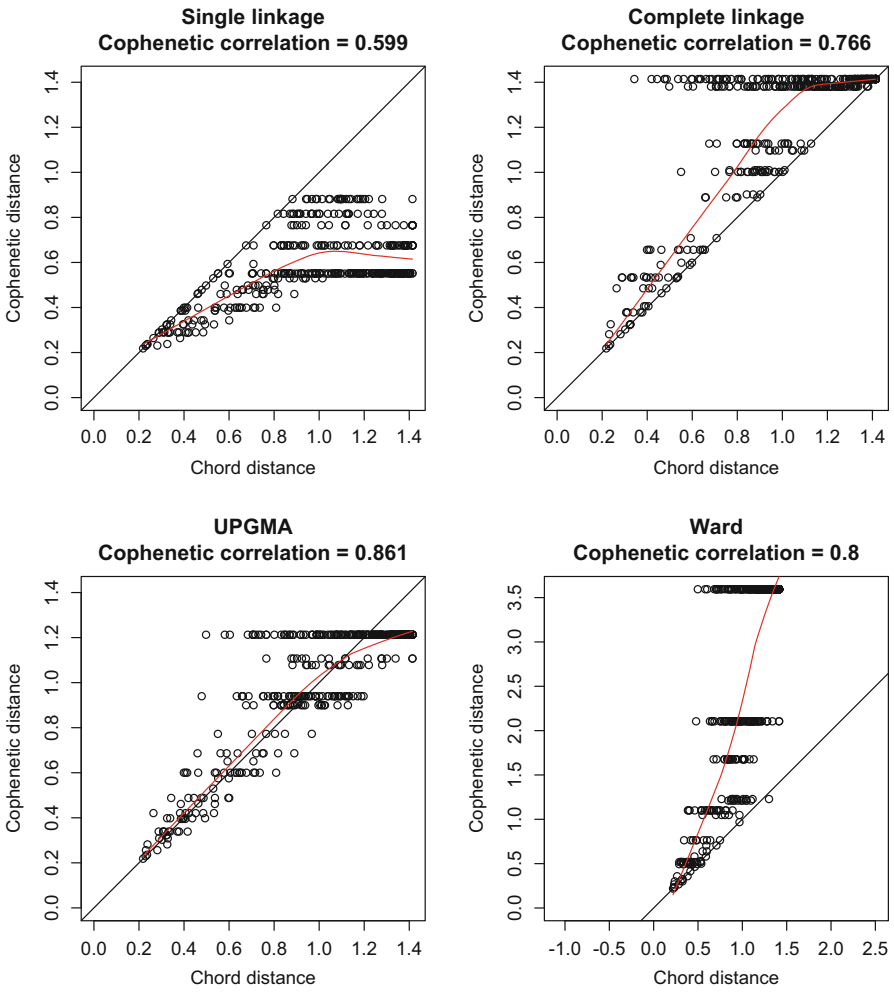


**Fig. 4.7** Shepard-like diagrams comparing chord distances (species data) to four cophenetic distances. A LOWESS smoother shows the trend in each plot

```
# Shepard-like diagrams
par(mfrow = c(2, 2))
plot(
  spe.ch,
  spe.ch.single.coph,
  xlab = "Chord distance",
  ylab = "Cophenetic distance",
  asp = 1,
  xlim = c(0, sqrt(2)),
  ylim = c(0, sqrt(2)),
  main = c("Single linkage", paste("Cophenetic correlation =",
           round(cor(spe.ch, spe.ch.single.coph), 3)))
)
abline(0, 1)
lines(lowess(spe.ch, spe.ch.single.coph), col = "red")
plot(
  spe.ch,
  spe.ch.comp.coph,
  xlab = "Chord distance",
  ylab = "Cophenetic distance",
  asp = 1,
  xlim = c(0, sqrt(2)),
  ylim = c(0, sqrt(2)),
  main = c("Complete linkage", paste("Cophenetic correlation =",
           round(cor(spe.ch, spe.ch.comp.coph), 3)))
)
abline(0, 1)
lines(lowess(spe.ch, spe.ch.comp.coph), col = "red")
plot(
  spe.ch,
  spe.ch.UPGMA.coph,
  xlab = "Chord distance",
  ylab = "Cophenetic distance",
  asp = 1,
  xlim = c(0, sqrt(2)),
  ylim = c(0, sqrt(2)),
  main = c("UPGMA", paste("Cophenetic correlation =",
           round(cor(spe.ch, spe.ch.UPGMA.coph), 3)))
)
abline(0, 1)
lines(lowess(spe.ch, spe.ch.UPGMA.coph), col = "red")
plot(
  spe.ch,
  spe.ch.ward.coph,
  xlab = "Chord distance",
  ylab = "Cophenetic distance",
  asp = 1,
  xlim = c(0, sqrt(2)),
  ylim = c(0, max(spe.ch.ward$height)),
  main = c("Ward", paste("Cophenetic correlation =",
           round(cor(spe.ch, spe.ch.ward.coph), 3)))
)
abline(0, 1)
lines(lowess(spe.ch, spe.ch.ward.coph), col = "red")
```

*Which method produces cophenetic distances (ordinate) that are well linearly related to the original distances (abscissa)?*

Another possible statistic for the comparison of clustering results is the Gower (1983) distance[1], computed as the sum of squared differences between the original dissimilarities and cophenetic distances. The clustering method that produces the smallest Gower distance may be seen as the one that provides the best clustering model of the dissimilarity matrix. The cophenetic correlation and Gower distance criteria do not always designate the same clustering result as the best.

```
# Gower (1983) distance
(gow.dist.single <- sum((spe.ch - spe.ch.single.coph) ^ 2))
(gow.dist.comp <- sum((spe.ch - spe.ch.comp.coph) ^ 2))
(gow.dist.UPGMA <- sum((spe.ch - spe.ch.UPGMA.coph) ^ 2))
(gow.dist.ward <- sum((spe.ch - spe.ch.ward.coph) ^ 2))
```

*Hint* *Enclosing in brackets a command line producing an object induces the immediate screen display of the object.*

### *4.7.3 Looking for Interpretable Clusters*

To interpret and compare clustering results, users generally look for interpretable clusters. This means that a decision must be made: at what level should the dendrogram be cut? Although it is not mandatory to select a single cutting level for a whole dendrogram (some parts of the dendrogram may be interpretable at finer levels than others), it is often practical to find one or a few levels where interpretations are made. These levels can be defined subjectively by visual examination of the dendrogram, or they can be chosen to fulfil some criteria, such as a predetermined number of groups for instance. In any case, adding information on the dendrograms or plotting additional information about the clustering results can be very useful.

#### 4.7.3.1 Graph of the Fusion Level Values

The fusion level values of a dendrogram are the dissimilarity values where a fusion between two branches of a dendrogram occurs. Plotting the fusion level values may

[1]This Gower distance is not to be confused with the Gower dissimilarity presented in Chap. 3.

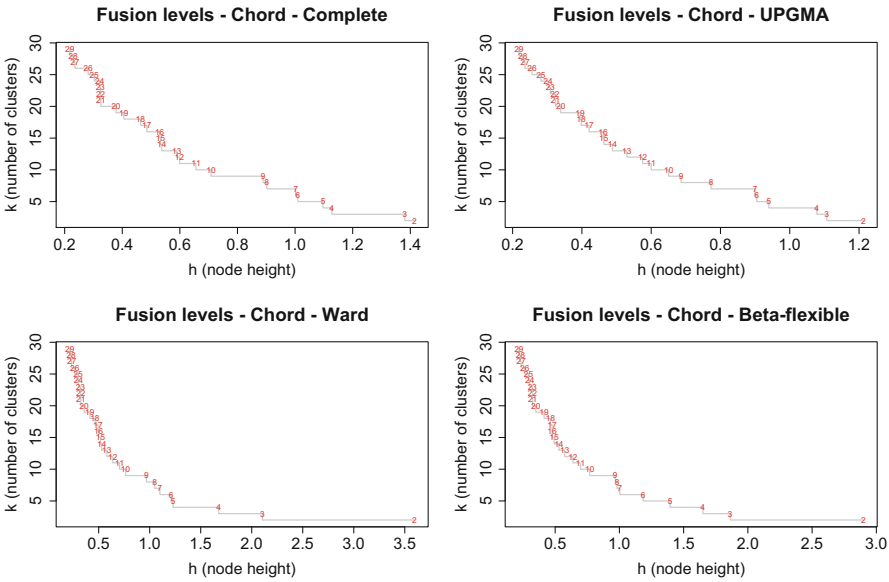


**Fig. 4.8** Graphs of the fusion level values of four dendrograms. When read from right to left, long horizontal lines preceding steep increases suggest cutting levels, e.g. the 9-group solution in the complete linkage dendrogram or the 4-group solution in the Ward dendrogram

help define cutting levels. Let us plot the fusion level values for some of the dendrograms produced above (Fig. 4.8).

```
par(mfrow = c(2, 2))
# Plot the fusion level values of the complete linkage clustering
plot(
  spe.ch.complete$height,
  nrow(spe):2,
  type = "S",
  main = "Fusion levels - Chord - Complete",
  ylab = "k (number of clusters)",
  xlab = "h (node height)",
  col = "grey"
)
text(spe.ch.complete$height,
     nrow(spe):2,
     nrow(spe):2,
     col = "red",
     cex = 0.8)
# Plot the fusion level values of the UPGMA clustering
plot(
  spe.ch.UPGMA$height,
  nrow(spe):2,
  type = "S",
  main = "Fusion levels - Chord - UPGMA",
  ylab = "k (number of clusters)",
  xlab = "h (node height)",
  col = "grey"
)
```

```
text(spe.ch.UPGMA$height,
     nrow(spe):2,
     nrow(spe):2,
     col = "red",
     cex = 0.8)
# Plot the fusion level values of the Ward clustering
plot(
  spe.ch.ward$height,
  nrow(spe):2,
  type = "S",
  main = "Fusion levels - Chord - Ward",
  ylab = "k (number of clusters)",
  xlab = "h (node height)",
  col = "grey"
)
text(spe.ch.ward$height,
     nrow(spe):2,
     nrow(spe):2,
     col = "red",
     cex = 0.8)
# Plot the fusion level values of the beta-flexible clustering
# (beta = -0.25)
plot(
  spe.ch.beta2$height,
  nrow(spe):2,
  type = "S",
  main = "Fusion levels - Chord - Beta-flexible",
  ylab = "k (number of clusters)",
  xlab = "h (node height)",
  col = "grey"
)
text(spe.ch.beta2$height,
     nrow(spe):2,
     nrow(spe):2,
     col = "red",
     cex = 0.8)
```

*Hint* *Observe how the function* **`text()`** *is used to print the number of groups (clusters) directly on the graph.*

*What is the suggested number of groups for each method? Go back to the dendrograms and cut them at the corresponding distances. Do the groups obtained always make sense? Do you obtain enough groups containing a substantial number of sites?*
*Remember that there is no single "truth" among these solutions. Each one may provide some insight into the data.*

As it is obvious from the dendrograms and the graphs of the fusion levels, the four analyses tell different stories.

Now, if you want to set a common number of groups and compare the group contents among dendrograms, you can use the **`cutree()`** function and compute contingency tables:

```
# Choose a common number of groups
k <- 4  # Number of groups where at least a small jump is present
        # in all four graphs of fusion levels
# Cut the dendrograms and create membership vectors
spech.single.g <- cutree(spe.ch.single, k = k)
spech.complete.g <- cutree(spe.ch.complete, k = k)
spech.UPGMA.g <- cutree(spe.ch.UPGMA, k = k)
spech.ward.g <- cutree(spe.ch.ward, k = k)
spech.beta.g <- cutree(spe.ch.beta2, k = k)

# Compare classifications by constructing contingency tables
# Single vs complete linkage
table(spech.single.g, spech.complete.g)
# Single linkage vs UPGMA
table(spech.single.g, spech.UPGMA.g)
# Single linkage vs Ward
table(spech.single.g, spech.ward.g)
# Complete linkage vs UPGMA
table(spech.complete.g, spech.UPGMA.g)
# Complete linkage vs Ward
table(spech.complete.g, spech.ward.g)
# UPGMA vs Ward
table(spech.UPGMA.g, spech.ward.g)
# beta-flexible vs Ward
table(spech.beta.g, spech.ward.g)
```

> *If two classifications had provided the same group contents, the contingency tables would have shown only one non-zero frequency value in each row and each column. This was almost never the case here. For instance, the 26 sites of the second group of the single linkage clustering are distributed over the four groups of the Ward clustering.*
> *However, one of the tables created above shows that two classifications resemble each other quite closely. Which ones?*

#### 4.7.3.2 Compare Two Dendrograms to Highlight Common Subtrees

To help select a partition found by several algorithms, it is useful to compare dendrograms and seek common clusters. The **`tangelgram()`** function from the **`dendextend`** package does this job nicely. Here we compare the dendrograms

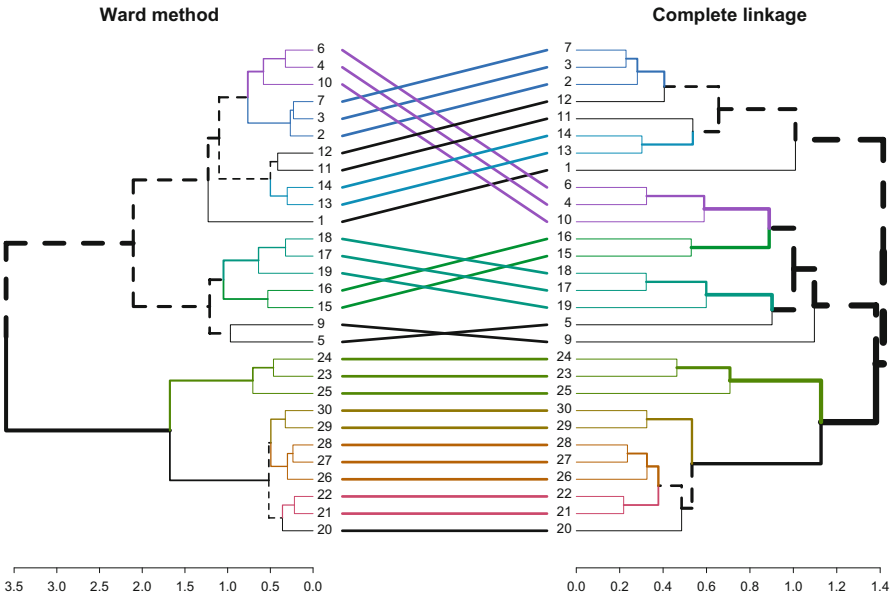


**Fig. 4.9** Pairwise comparison of two dendrograms

previously produced by the Ward and complete linkage clustering methods (Fig. 4.9):

```
# Objects of class "hclust" must be first converted into objects of
# class "dendrogram"
class(spe.ch.ward)     # [1] "hclust"
dend1 <- as.dendrogram(spe.ch.ward)
class(dend1)           # [1] "dendrogram"
dend2 <- as.dendrogram(spe.ch.complete)
dend12 <- dendlist(dend1, dend2)
tanglegram(
  untangle(dend12),
  sort = TRUE,
  common_subtrees_color_branches = TRUE,
  main_left = "Ward method",
  main_right = "Complete linkage"
)
```

*Hint* *In order for the* **`tanglegram()`** *result to display the site names (as opposed to their number in the data set), the code in Sect. 4.3.1 shows how to incorporate the site labels into an object of class "dist" (here* **`spe.ch`***) used to compute the clusterings, by means of function* **`attr()`**.

*Sites are ordered to match the two dendrogramsas well as possible. Colours highlight common clusters, whereas sites printed in black have different positions in the two trees. Can you recognize particularly "robust" clusters?*

#### 4.7.3.3 Multiscale Bootstrap Resampling

Unconstrained cluster analysis is a family of heuristic methods that are not aimed at testing *a priori* hypotheses. However, natural variation leads to sampling variability, and the results of a classification are likely to reflect it. It is therefore legitimate to assess the uncertainty (or its counterpart the robustness) of a classification. This has been done abundantly in phylogenetic analysis.

The choice approach for such validation procedures is bootstrap resampling (e.g. Efron 1979; Felsenstein 1985), which consists in randomly sampling subsets of the data and computing the clustering on these subsets. After having done this a large number of times, one counts the proportion of the replicate clustering results where a given cluster appears. This proportion is called the bootstrap probability (BP) of the cluster. Multiscale bootstrap resampling has been developed as an enhancement to answer some criticisms about the classical bootstrap procedure (Efron et al. 1996; Shimodaira 2002, 2004). In this method bootstrap samples of several different sizes are used to estimate the p-value of each cluster. This improvement produces "approximately unbiased" (AU) p-values. Readers are referred to the original publications for more details.

The **`pvclust`** package (Suzuki and Shimodaira 2006) provides functions to plot a dendrogram with bootstrap p-values associated to each cluster. AU p-values are printed in red. The less accurate BP values are printed in green. Clusters with high AU values (e.g. 0.95 or more) can be considered as strongly supported by the data. Let us apply this analysis to the dendrogram obtained with the Ward method (Fig. 4.10). **Beware**: in function **`pvclust()`** the data object must be transposed with respect to our usual layout (rows are variables).

```
# Compute p-values for all clusters (edges) of the dendrogram
spech.pv <-
  pvclust(t(spe.norm),
          method.hclust = "ward.D2",
          method.dist = "euc",
          parallel = TRUE)

# Plot dendrogram with p-values
plot(spech.pv)
# Highlight clusters with high AU p-values
pvrect(spech.pv, alpha = 0.95, pv = "au")
lines(spech.pv)
pvrect(spech.pv, alpha = 0.91, border = 4)
```

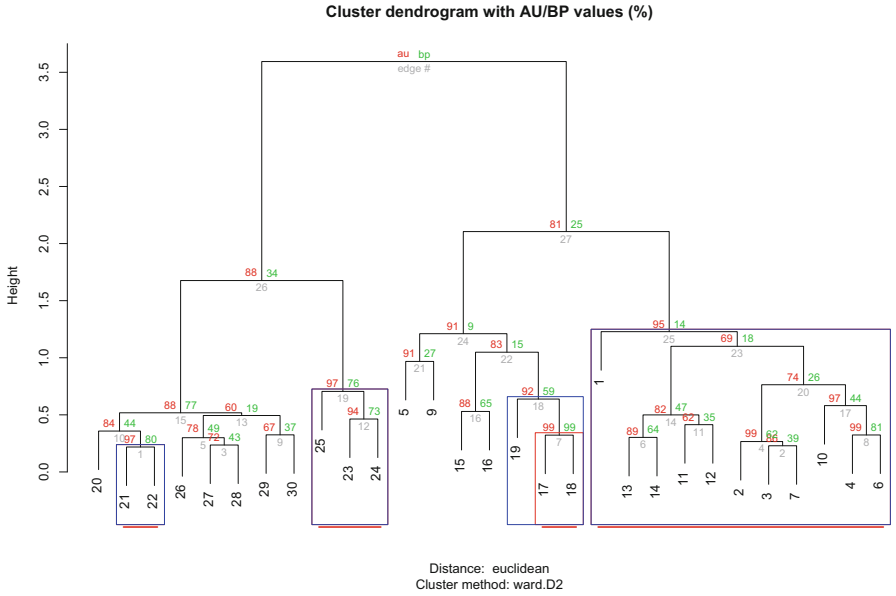


**Fig. 4.10** Multiscale bootstrap resampling applied to the Chord-Ward dendrogram

*Hint* *The argument* `parallel = TRUE` *is used to greatly accelerate the computation with large data sets. It calls upon package* **`parallel`**.

*Clusters enclosed in red boxes and underlined correspond to significant AU p-values ($p \geq 0.95$), whereas clusters enclosed in blue rectangles show less significant values ($p \geq 0.91$ in this example). This analysis is useful to highlight the most "robust" groups of sites. Note, however, that for the same data the results may vary from run to run because this is a procedure based on random resampling*

Once we have chosen a dendrogram and assessed the robustness of its clusters, let us examine three methods that can help us identify an appropriate number of groups: silhouette widths, matrix comparison and diagnostic species.

#### 4.7.3.4 Average Silhouette Widths

The *silhouette width* is a measure of the degree of membership of an object to its cluster, based on the average dissimilarity between this object and all objects of the

cluster to which it belongs, compared to the same measure computed for the next closest cluster (see Sect. 4.7.3.7). Silhouette widths range from −1 to 1 and can be averaged over all objects of a partition.

We shall use the function **`silhouette()`** of package **`cluster`**. The documentation file of this function provides a formal definition of a silhouette width. In short, the larger the value is, the better the object is clustered. Negative values suggest that the corresponding objects may have been placed in the wrong cluster.

At each fusion level, the average silhouette width can be used as a measure of the quality of the partition (Rousseeuw quality index): compute silhouette widths measuring the intensity of the link of the objects to their groups, and choose the level where the within-group mean intensity is the highest, that is, the largest average silhouette width. A barplot is drawn (Fig. 4.11).

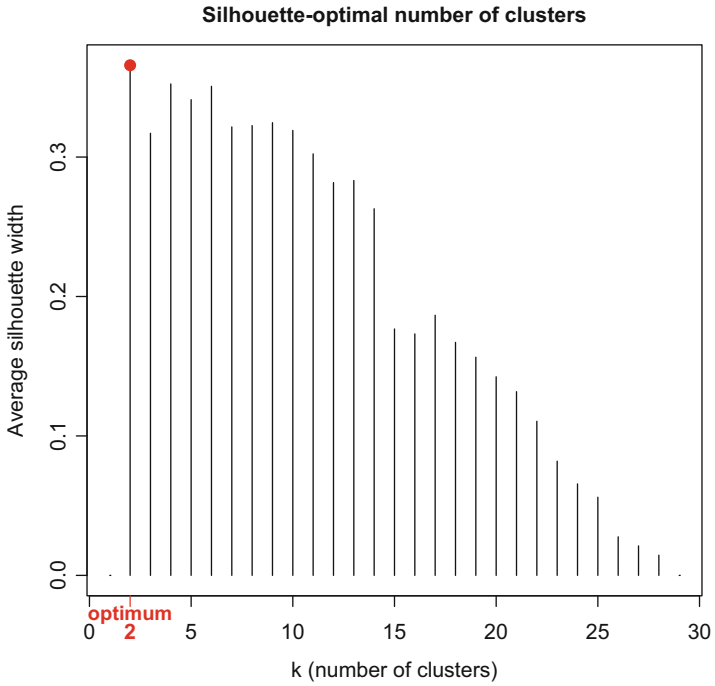


**Fig. 4.11** Barplot showing the average silhouette widths for $k = 2$ to 29 groups. The best partition by this criterion is the one with the largest average silhouette width, i.e., in two groups. The second best, in 4 groups, and the third best, in 6 groups, are more interesting from an ecological point of view

```
# Choose and rename the dendrogram ("hclust" object)
hc <- spe.ch.ward

# Plot average silhouette widths (using Ward clustering) for all
# partitions except for the trivial partitions (k = 1 or k = n)
Si <- numeric(nrow(spe))
for (k in 2:(nrow(spe) - 1))
{
  sil <- silhouette(cutree(hc, k = k), spe.ch)
  Si[k] <- summary(sil)$avg.width
}
k.best <- which.max(Si)
plot(
  1:nrow(spe),
  Si,
  type = "h",
  main = "Silhouette-optimal number of clusters",
  xlab = "k (number of clusters)",
  ylab = "Average silhouette width"
)
axis(
  1,
  k.best,
  paste("optimum", k.best, sep = "\n"),
  col = "red",
  font = 2,
  col.axis = "red"
)
points(k.best,
       max(Si),
       pch = 16,
       col = "red",
       cex = 1.5)
```

*Hint* *Observe how the repeated computation of the average silhouette widths is done by a* **`for()`** *loop.*

*As it often happens, this criterion has selected two groups as the optimal number. Considering the site numbers (reflecting their location along the river) in the dendrogram (Fig. 4.10), what is the explanation of this partitionin two groups? Is it interesting from an ecological point of view?*

#### 4.7.3.5 Comparison Between the Dissimilarity Matrix and Binary Matrices Representing Partitions

This technique compares the original dissimilarity matrix to binary matrices computed from the dendrogram cut at various levels (and representing group allocations). The idea is to choose the level where the matrix correlation between the two is the highest. Tests are impossible here since the matrices are not independent of one another; indeed, the matrices corresponding to the partitions are all derived from the original dissimilarity matrix.

To compute the binary dissimilarity matrices representing group membership, we will use our homemade function **`grpdist()`** to compute a binary dissimilarity matrix from a vector defining groups. The results are shown in Fig. 4.12.

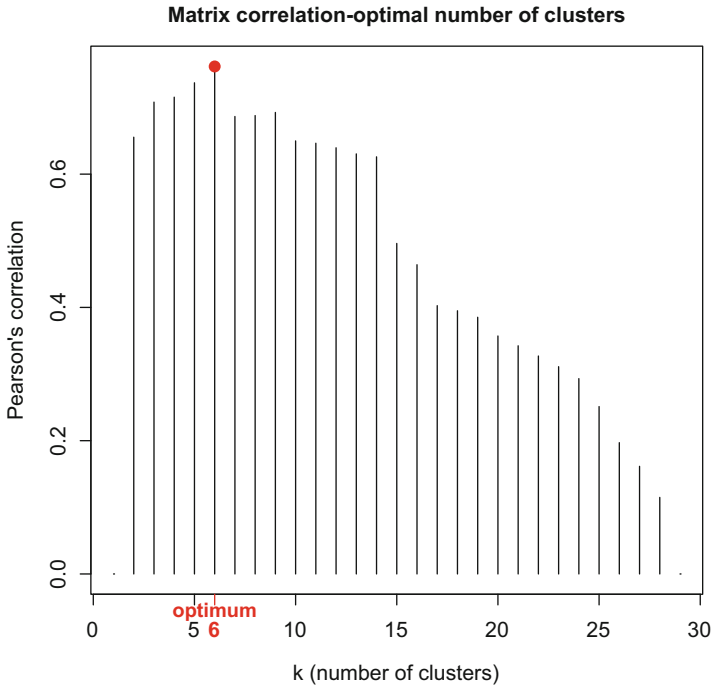


**Fig. 4.12** Barplot showing the matrix correlations between the original dissimilarity matrix and binary matrices computed from the dendrogram cut at various levels

```
# Optimal number of clusters according to matrix correlation
# statistic (Pearson)
kt <- data.frame(k = 1:nrow(spe), r = 0)
for (i in 2:(nrow(spe) - 1)) {
  gr <- cutree(hc, i)
  distgr <- grpdist(gr)
  mt <- cor(spe.ch, distgr, method = "pearson")
  kt[i, 2] <- mt
}
k.best <- which.max(kt$r)
plot(
  kt$k,
  kt$r,
  type = "h",
  main = "Matrix correlation-optimal number of clusters",
  xlab = "k (number of clusters)",
  ylab = "Pearson's correlation"
)
axis(
  1,
  k.best,
  paste("optimum", k.best, sep = "\n"),
  col = "red",
  font = 2,
  col.axis = "red"
)
points(k.best,
       max(kt$r),
       pch = 16,
       col = "red",
       cex = 1.5)
```

*The barplot shows that a partition in 3 to 6 clusters would achieve a high matrix correlation between the chord distance matrix and the binary allocation matrix.*

#### 4.7.3.6 Species Fidelity Analysis

Another internal criterion for assessing the quality of a partition is based on species fidelity analysis. The basic idea is to retain clusters that are best characterized by a set of diagnostic species, also called "indicator", "typical", "characteristic" or "differential" species, i.e. species that are significantly more frequent and abundant in a given group of sites. Specifically, the best partition would be the one that maximizes both (i) the sum of indicator values and (ii) the proportion of clusters with significant indicator species. Here, we shall anticipate on Sect. 4.11.2 and use index IndVal (Dufrêne and Legendre 1997), which integrates a specificity and a fidelity measure (Fig. 4.13).

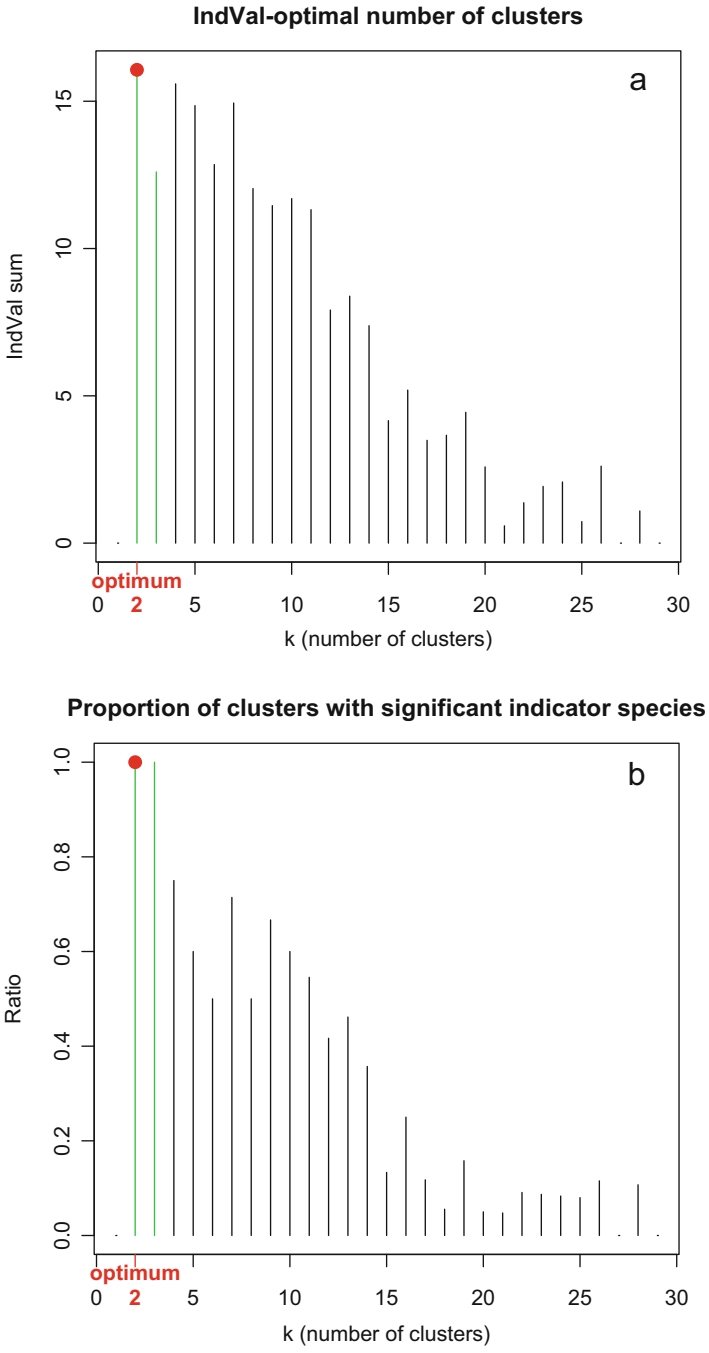


**Fig. 4.13** Barplots showing the sum of species indicator values (IndVal) (**a**) and the proportion of significant indicator species (**b**) for all partitions obtained after cutting the dendrogram at various levels. The solutions where there are significant indicator species in all *k* clusters are highlighted in green

```
# Optimal number of clusters as per indicator species analysis
# (IndVal, Dufrene-Legendre; package: labdsv)
IndVal <- numeric(nrow(spe))
ng <- numeric(nrow(spe))
for (k in 2:(nrow(spe) - 1)) {
  iva <- indval(spe, cutree(hc, k = k), numitr = 1000)
  gr <- factor(iva$maxcls[iva$pval <= 0.05])
  ng[k] <- length(levels(gr)) / k
  iv <- iva$indcls[iva$pval <= 0.05]
  IndVal[k] <- sum(iv)
}
k.best <- which.max(IndVal[ng == 1]) + 1
col3 <- rep(1, nrow(spe))
col3[ng == 1] <- 3

par(mfrow = c(1, 2))
plot(
  1:nrow(spe),
  IndVal,
  type = "h",
  main = "IndVal-optimal number of clusters",
  xlab = "k (number of clusters)",
  ylab = "IndVal sum",
  col = col3
)
axis(
  1,
  k.best,
  paste("optimum", k.best, sep = "\n"),
  col = "red",
  font = 2,
  col.axis = "red"
)
points(
  which.max(IndVal),
  max(IndVal),
  pch = 16,
  col = "red",
  cex = 1.5
)
plot(
  1:nrow(spe),
  ng,
  type = "h",
  xlab = "k (number of groups)",
  ylab = "Ratio",
  main = "Proportion of clusters with significant indicator
          species",
  col = col3
)
```

```
axis(1,
     k.best,
     paste("optimum", k.best, sep = "\n"),
     col = "red",
     col.axis = "red")
points(k.best,
       max(ng),
       pch = 16,
       col = "red",
       cex = 1.5)
```

*Hint* *In addition to the search for the solutions meeting the two criteria, the code above highlights in green the solutions where there are significant indicator species in the* k *clusters (object* `ng`*). In the Doubs data set, it is the case only in the 2- and 3-cluster solutions.*

> *Only the partition in two clusters, simply contrasting the upstream and downstream sites, fully meets the two criteria. However, a partition in three or four clusters, allowing the discovery of more subtle structures, would be an acceptable choice despite the absence of positive differential species (i.e., a species that is more frequent in a given group than in others) for some groups.*

#### 4.7.3.7 Silhouette Plot of the Final Partition

In our example, the silhouette-based, matrix correlation-based and IndVal-based criteria do not return the same solution; these are, ranging from $k = 2$ to $k = 6$. A good compromise seems to be $k = 4$. Let us select this number for our final group diagnostics. We can select the Ward clustering as our final choice, since this method produced four well-balanced (not equal-sized, but without outliers) and well-delimited groups.

We can now proceed to examine if the group memberships are appropriate (i.e., no or few objects apparently misclassified). A silhouette plot is useful here (Fig. 4.14).

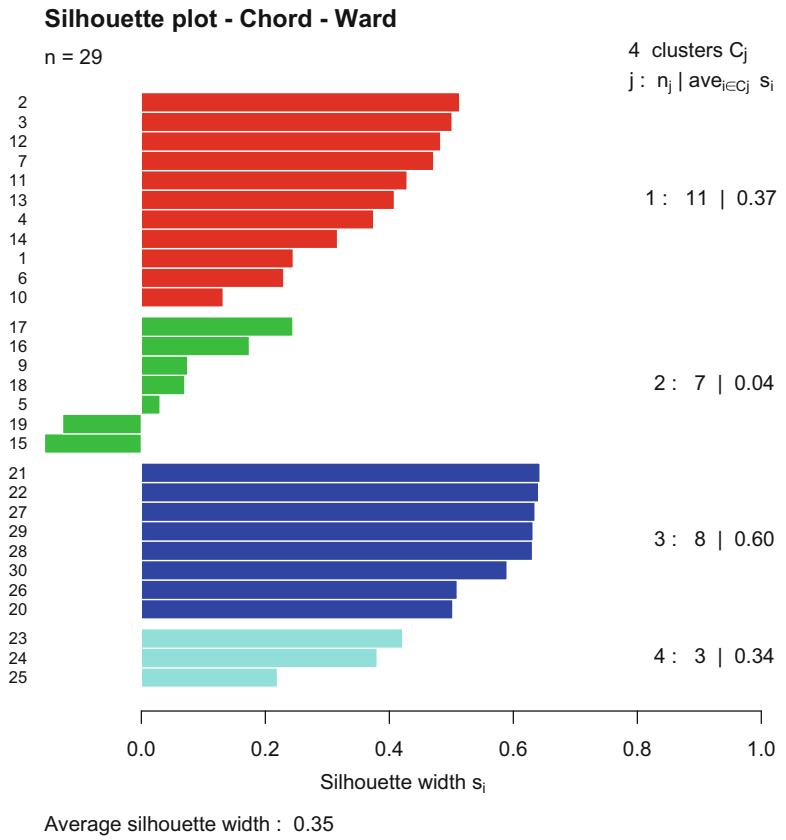


**Fig. 4.14** Silhouette plot of the final, four-group partition from Ward clustering

```
# Choose the number of clusters
k <- 4
# Silhouette plot of the final partition
spech.ward.g <- cutree(spe.ch.ward, k = k)
sil <- silhouette(spech.ward.g, spe.ch)
rownames(sil) <- row.names(spe)
plot(
  sil,
  main = "Silhouette plot - Chord - Ward",
  cex.names = 0.8,
  col = 2:(k + 1),
  nmax = 100
)
```

*Clusters 1 and 3 are the most coherent, while cluster 2 contains misclassified objects.*

#### 4.7.3.8 Final Dendrogram with Graphical Options

Now it is time to produce the final dendrogram, where we can represent the four groups and improve the overall appearance with several graphical options (Fig. 4.15). Try the code, compare the results with it, and use the documentation files to understand the arguments used. Note also the use of a homemade function **`hcoplot()`**.

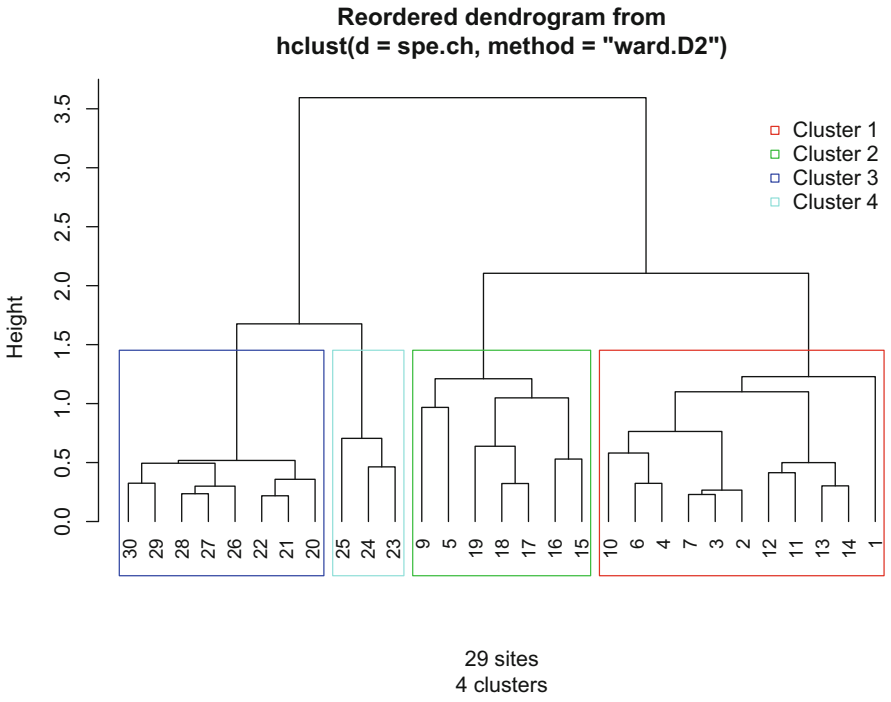


**Fig. 4.15** Final dendrogram with boxes around the four selected clusters of sites from the fish species data

```
# Reorder clusters
spe.chwo <- reorder.hclust(spe.ch.ward, spe.ch)

# Plot reordered dendrogram with group labels
plot(
  spe.chwo,
  hang = -1,
  xlab = "4 groups",
  sub = "",
  ylab = "Height",
  main = "Chord - Ward (reordered)",
  labels = cutree(spe.chwo, k = k)
)
rect.hclust(spe.chwo, k = k)

# Plot the final dendrogram with group colors (RGBCMY...)
# Fast method using the additional hcoplot() function:
hcoplot(spe.ch.ward, spe.ch, lab = rownames(spe), k = 4)
```

*Hints* *Function* **reorder.hclust()** *reorders objects so that their original order in the dissimilarity matrix is respected as much as possible. This does not affect the topology of the dendrogram. Its arguments are (1) the clustering object and (2) the dissimilarity matrix.*

*When using* **rect.hclust()** *to draw boxes around clusters, as in the* **hcoplot()** *function used here, one can specify a fusion level (argument* `h`*) instead of a number of groups (argument* `k`*).*

*Another function called* **identify.hclust()** *allows the interactive cut of a tree at any position where one left-clicks with the mouse. It makes it possible to extract a list of objects from any given subgroup.*

*The argument* `hang = -1` *specifies that the branches of the dendrogram will all reach the value 0 and the labels will hang below that value.*

Let us conclude with several other representations of the clustering results obtained above. The usefulness of these representations depends on the context.

The **dendextend** package provides various functions to improve the representation of a dendrogram. First, the `hclust` object must be converted to a `dendrogram` object. Then, you can apply colours and line options to the branches of the dendrogram, e.g. based on the final partition (Fig. 4.16).

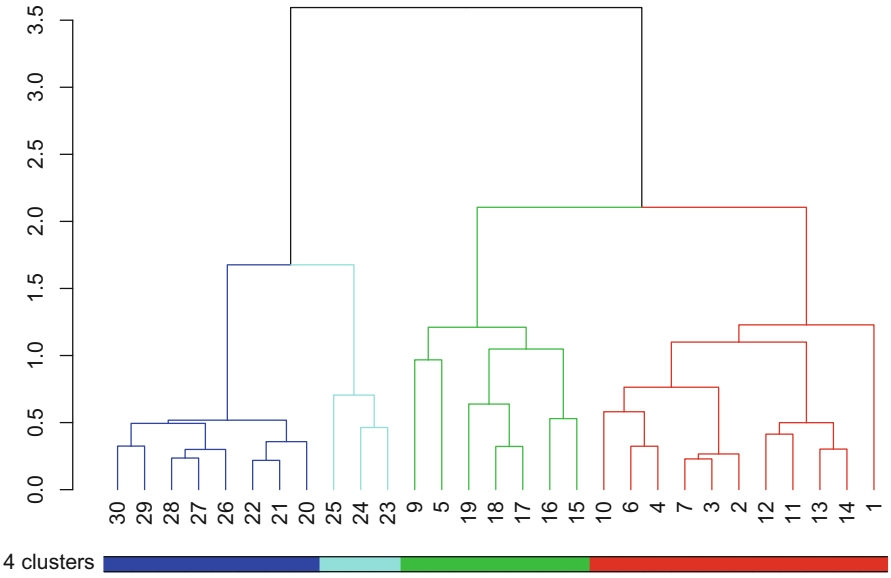


**Fig. 4.16** Final dendrogram with coloured branches highlighting the four selected clusters of sites from the fish species data

```
# Convert the "hclust" object into a "dendrogram" object
dend <- as.dendrogram(spe.chwo)

# Plot the dendrogram with coloured branches
dend %>% set("branches_k_color", k = k) %>% plot

# Use standard colours for clusters
clusters <- cutree(dend, k)[order.dendrogram(dend)]
dend %>%
set("branches_k_color", k = k, value = unique(clusters) + 1) %>%
plot
# Add a coloured bar
colored_bars(clusters + 1,
             y_shift = -0.5,
             rowLabels = paste(k, "clusters"))
```

*Hint* *Note the non-conventional syntax used by several recent packages, including* **dendextend**, *based on a chaining of instructions separated by the operator* %>%.

### 4.7.3.9 Spatial Plot of the Clustering Result

The following code allows us to plot the clusters on a map representing the river. This is a very useful way of representing results for spatially explicit data (Fig. 4.17).

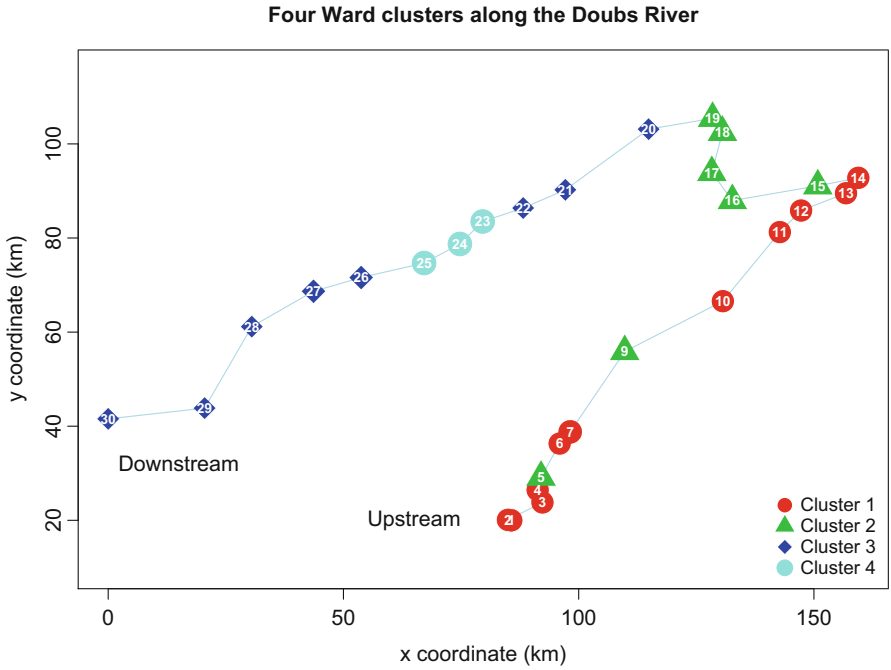


**Fig. 4.17** The four ward clusters on a map of the Doubs River. Site 1 is hidden underneath site 2

Since we shall draw comparable maps later on, we assemble the code in a homemade generic function called **`drawmap()`**:

```
# Plot the Ward clusters on a map of the Doubs River
# (see Chapter 2)
drawmap(xy = spa,
        clusters = spech.ward.g,
        main = "Four Ward clusters along the Doubs River")
```

> *Go back to the maps of four fish species (Chap.2). Compare these to the map created above and shown in Fig. 4.17.*

#### 4.7.3.10 Heat Map and Ordered Community Table

See how to represent the dendrogram in a square matrix of coloured pixels, where the colour intensity represents the similarity among the sites (Fig. 4.18):

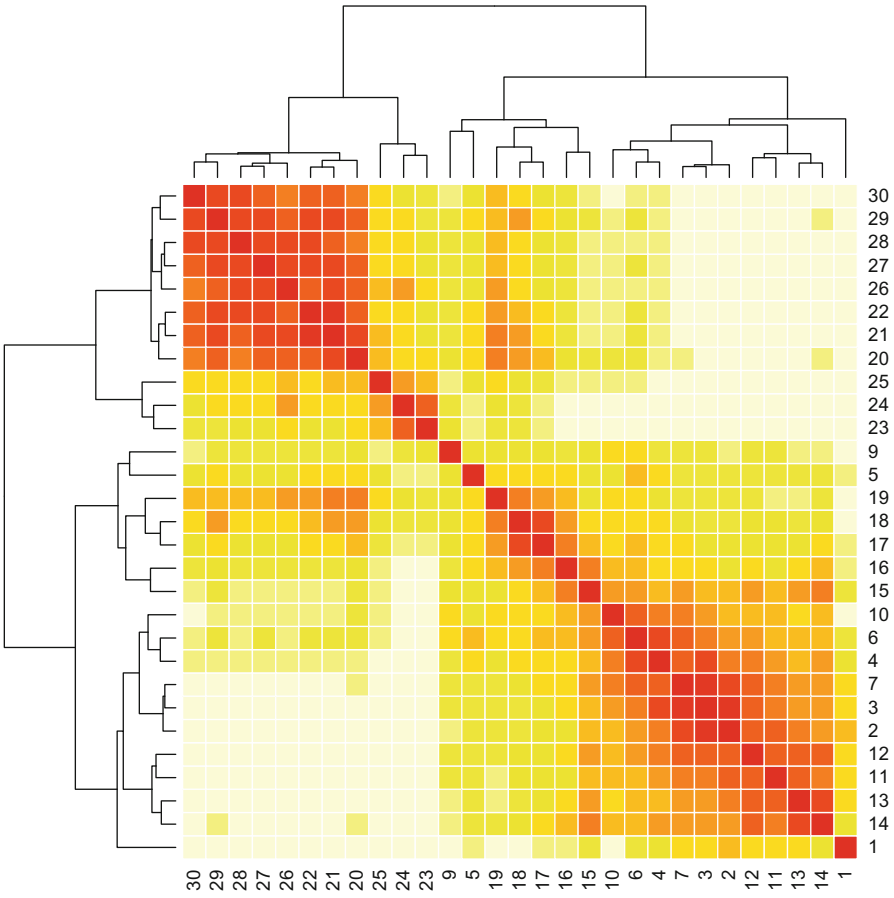


**Fig. 4.18** Heat map of the chord *D* matrix reordered following the dendrogram

```
# Heat map of the dissimilarity matrix ordered with the dendrogram
heatmap(
  as.matrix(spe.ch),
  Rowv = dend,
  symm = TRUE,
  margin = c(3, 3)
)
```

*Hint* *Note that the* `hclust` *object has been converted to an object of class* `dendrogram`*. This allows many advanced graphical manipulations of tree-like structures. See the documentation file of the* **`as.dendrogram()`** *function, where examples are given.*

*Observe how, thanks to the ordering, most "hot" (red) values representing high similarities are located close to the diagonal. The diagonal itself is trivial.*

Finally, it may be useful to explore the clusters' species contents directly, i.e. to reorder the original data table according to the group memberships. In order to avoid large values to clog the picture, Jari Oksanen proposes **`vegemite()`**, a function of **`vegan`** that can use external information to reorder and display a site-by-species data table where abundance values can be recoded in different ways. If no specific order is provided for the species, these are ordered by their weighted averages on the site scores. Note that the abundance scale accepted by **`vegemite()`** must be made of one-digit counts only (e.g. from 0 to 9). Any two- or more digit value will cause **`vegemite()`** to stop. Our abundance data, expressed on a 0–5 scale, cause no problem. Other data may have to be recoded. **`vegemite()`** itself makes an internal use of another function, **`coverscale()`**, devoted to the rescaling of vegetation data.

```
# Ordered community table
# Species are ordered by their weighted averages on site scores.
# Dots represent absences.
or <- vegemite(spe, spe.chwo)
```

```
      32222222222  111111     1111
      09876210543959876506473221341
Icme  5432121.......................
Abbr  54332431.....1................
Blbj  54542432.1...1................
Anan  54432222.....111..............
Gyce  5555443212...11...............
Scer  522112221...21................
Cyca  53421321.....1111.............
Rham  55432333.....221..............
Legi  35432322.1...1111.............
Alal  55555555352..322..............
Chna  12111322.1...211..............
Titi  53453444...1321111.21.........
Ruru  55554555121455221..1.........
Albi  53111123.....2341.............
Baba  35342544.....23322.........1.
Eslu  453423321...41111..12.1....1.
Gogo  5544355421..242122111......1.
Pefl  54211432....41321..12.........
Pato  2211.222.....3344.............
Sqce  3443242312152132232211..11.1.
Lele  332213221...52235321.1.......
Babl  .1111112...32534554555534124.
Teso  .1...........11254........23.
Phph  .1....11...13334344454544455.
Cogo  ..............1123......2123.
Satr  .1..........2.123413455553553
Thth  .1............11.2......2134.
sites species
   29      27
```

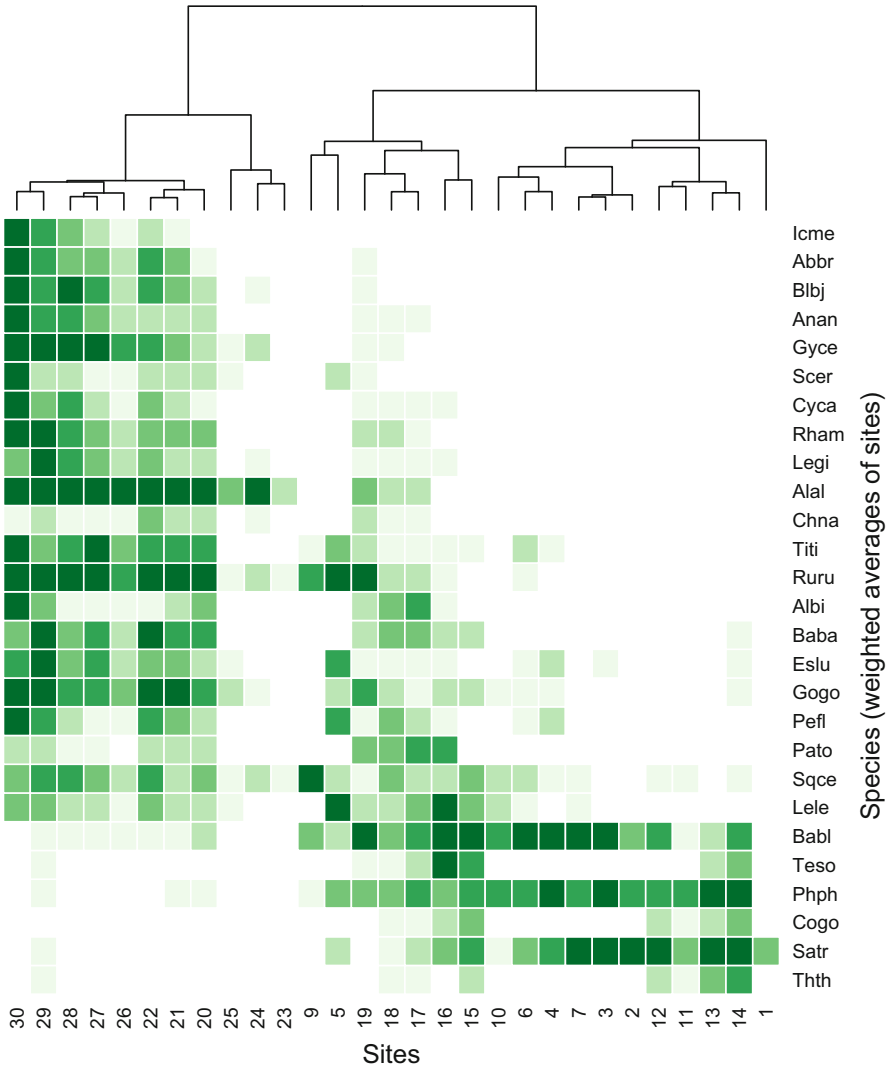


**Fig. 4.19** Heat map of the doubly ordered community table, with dendrogram

This way of ordering species can also be used in a heat map with colour intensities proportional to the species abundances (Fig. 4.19):

```
# Heat map of the doubly ordered community table, with dendrogram
heatmap(
  t(spe[rev(or$species)]),
  Rowv = NA,
  Colv = dend,
  col = c("white", brewer.pal(5, "Greens")),
  scale = "none",
  margin = c(4, 4),
  ylab = "Species (weighted averages of sites)",
  xlab = "Sites"
)
```

*Hint* *A similar result can be obtained using the* **tabasco()** *function of the* **vegan** *package.*

## 4.8 Non-hierarchical Clustering

Non-hierarchical partitioning consists in looking for a single partition of a set of objects. The problem can be stated as follows: given $n$ objects in a $p$-dimensional space, determine a partition of the objects into $k$ groups, or clusters, such that the objects within each cluster are more similar to one another than to objects in the other clusters. The user determines the number of groups, $k$. The partitioning algorithms require an initial configuration, i.e. an initial attribution of the objects to the $k$ groups, which will be optimized in a recursive process. The initial configuration may be provided by theory, but users often choose a random starting configuration. In that case, the analysis is run a large number of times with different random initial configurations in the hope to find the best solution.

Here we shall present two related methods, $k$-means partitioning and partitioning around medoids (PAM). These algorithms operate in Euclidean space. An important note is that if the variables in the data table are not dimensionally homogeneous, they must be standardized prior to partitioning. Otherwise, the total variance of the data has dimension equal to the sum of the squared dimensions of the individual variables, which is meaningless.

### *4.8.1* k-*means Partitioning*

The $k$-means method uses the local structure of the data to delineate clusters: groups are formed by identifying high-density regions in the data. To achieve this, the method iteratively minimizes an objective function called the total error sum of squares ($E^2_k$ or TESS or SSE), which is the sum of the within-group sums-of-squares. This quantity is the sum, over the $k$ groups, of the sums of the squared (Euclidean) distances among the objects in the groups, each divided by the number

of objects in the group. This is the same criterion as used in Ward's agglomerative clustering.

#### 4.8.1.1 *k*-means with Random Starts

If one has a pre-determined number of groups in mind, the recommended function to use is **`kmeans()`** of the **`stats`** package. The analysis can be automatically repeated a large number of times (argument `nstart`) using different random initial configurations. The function finds the best solution (smallest SSE value) after repeating the analysis 'nstart' times.

*k*-means is a *linear* method, i.e. it is not appropriate for raw species abundance data with lots of zeros (see Sect. 3.2.2). One possibility is to use non-Euclidean dissimilarity matrices, like the percentage difference (aka Bray-Curtis); such matrices should be square-root transformed and submitted to principal coordinate analysis (PCoA, Sect. 5.5) in order to obtain a full representation of the objects in Euclidean space. The resulting PCoA axes are then subjected to *k*-means partitioning. Another solution is to pre-transform the species data. To remain coherent with the previous sections, we can use the chord-transformed or "normalized" species data created in Sect. 4.3.1. When applied in combination with the Euclidean distance implicit in *k*-means, the analysis preserves the chord distance among sites. To compare with the results of Ward's clustering computed above, we ask for $k = 4$ groups and compare the outcome with the four groups derived from the Ward hierarchical clustering.

```
# k-means partitioning of the pre-transformed species data
# With 4 groups
spe.kmeans <- kmeans(spe.norm, centers = 4, nstart = 100)
spe.kmeans
```

> *Note: running the function again may produce slightly different results as each of the 'nstart' runs starts with a different random configuration. In particular, numbers given by the function to clusters are arbitrary!*

```
# Comparison with the 4-group partition derived from
# Ward clustering
table(spe.kmeans$cluster, spech.ward.g)
```

```
   spech.ward.g
     1  2  3  4
1    0  0  0  3
2   11  1  0  0
3    0  0  8  0
4    0  6  0  0
```

*Are the two results fairly similar? Which object(s) is (or are) classified differently?*

To compute *k*-means partitioning from dissimilarity indices that cannot be obtained by a transformation of the raw data followed by calculation of the Euclidean distance, for example the percentage difference (aka Bray-Curtis) dissimilarity, one has to compute first a rectangular data table with *n* rows by principal coordinate analysis (PCoA, Sect. 5.5) of the dissimilarity matrix, then use that rectangular table as input to *k*-means partitioning. For the percentage difference dissimilarity, one has to compute the PCoA on the square root of the percentage difference dissimilarities to obtain a fully Euclidean solution, or use a PCoA function that provides a correction for negative eigenvalues. These points are discussed in Chap. 5.

A partitioning yields a single partition with a predefined number of groups. If you want to try several solutions with different *k* values, you must rerun the analysis. But which solution is the best in terms of number of clusters? To answer this question, one has to state what "best" means. Many criteria exist; some of them are available in the function **`clustIndex()`** of the package **`cclust`**. Milligan and Cooper (1985) recommend maximizing the Calinski-Harabasz index (*F*-statistic comparing the among-group to the within-group sum of squares of the partition), although its value tends to be lower for unequal-sized partitions. The maximum of 'ssi' ("Simple Structure Index", see the documentation file of **`clustIndex()`** for details) is another good indicator of the best partition in the least-squares sense.

Fortunately, one can avoid running **`kmeans()`** many times by hand. **`vegan`**'s function **`cascadeKM()`** is a *wrapper* for the **`kmeans()`** function, that is, a function that uses a basic function, adding new properties to it. It creates several partitions forming a cascade from small (argument **`inf.gr`**) to large values of *k* (argument **`sup.gr`**). Let us apply this function to our dataset, asking for 2–10 groups and the simple structure index criterion for clustering quality, followed by a plot of the results (Fig. 4.20).

```
# k-means partitioning, 2 to 10 groups
spe.KM.cascade <-
  cascadeKM(
    spe.norm,
    inf.gr = 2,
    sup.gr = 10,
    iter = 100,
    criterion = "ssi"
  )
summary(spe.KM.cascade)
plot(spe.KM.cascade, sortg = TRUE)
```

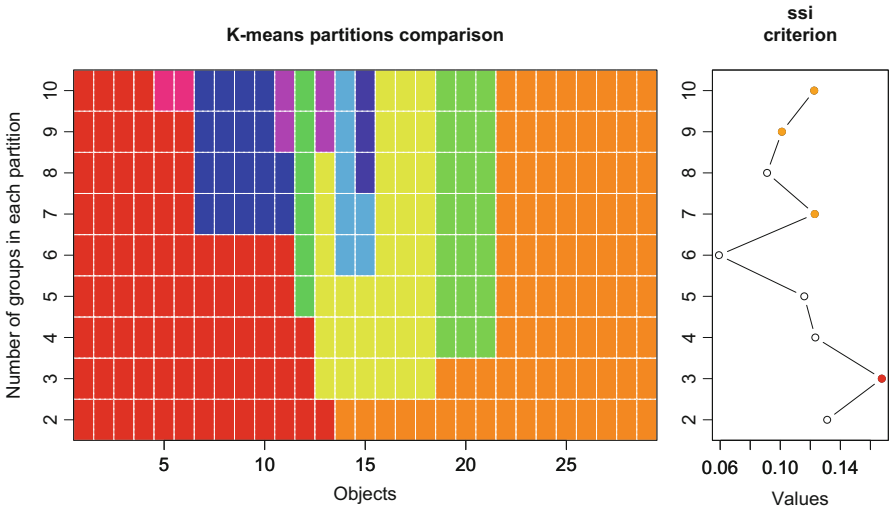


**Fig. 4.20** *k*-means cascade plot showing the group attributed to each object for each partition

*Hint* *In the plot,* `sortg = TRUE` *reorders the objects in such a way as to put together, insofar as possible, the objects pertaining to each group. A more detailed explanation of this argument in provided in the function's documentation file.*

*The plot shows the group attributed to each object for each partition (rows of the graph). The rows of the graph are the different values of k. The groups are represented by different colours; there are two colours for k = 2, three colours for k = 3, and so on. Another graph shows the values of the chosen stopping criterion for the different values of k. Since this is an iterative process, the results can vary from run to run.*
*How many groups does this cascade propose as the best solution? If one has reasons to prefer a larger number of groups, what would be the next best solution?*

The function **`cascadeKM()`** provides numeric results as well. Among them, the element 'result' gives the TESS statistic and the value of the criterion (`calinski` or `ssi`) for each value of *k*. The element 'partition' contains a table showing the group attributed to each object. If the geographic coordinates of the objects are available, they can be used to plot a map of the objects, with symbols or colours representing the groups specified by one of the columns of this table.

```
summary(spe.KM.cascade)
spe.KM.cascade$results
```

> *The minimum of SSE is the criterion used by the algorithm to find the optimal grouping of the objects for a given value of k, while calinski and ssiare good criteria to find the optimal value of k.*
> *Remember that the different partitions in k = {2, 3, ..., 10} groups are computed independently of one another. Examining the plot from bottom to top is NOT equivalent to examining a dendrogram because groups of successive partitions are not necessarily nested.*

After defining site clusters, it is time to examine their contents. The simplest way is to define subgroups of sites on the basis of the typology retained and compute basic statistics. You can run the following example based upon the *k*-means 4-group partition.

```
# Reorder the sites according to the k-means result
spe.kmeans.g <- spe.kmeans$cluster
spe[order(spe.kmeans.g), ]

# Reorder sites and species using function vegemite()
ord.KM <- vegemite(spe, spe.kmeans.g)
spe[ord.KM$sites, ord.KM$species]
```

#### 4.8.1.2 Use of *k*-means Partitioning to Optimize an Independently Obtained Classification

From another point of view, *k*-means partitioning can be used to optimise the result of a hierarchical clustering. Indeed, the nature of an agglomeration algorithm such as those explored in the previous sections prevents an object that has been included in a group to be translocated to another that appeared later in the agglomeration process, even if the latter would now be more appropriate. To overcome this potential problem, one can provide the *k*-means algorithm with prior information derived from the clustering to be optimized, either in the form of a $k \times p$ matrix of mean values of the $p$ variables in the $k$ groups obtained with the method, or as a list of $k$ objects, one per group, considered to be "typical" for each group. In the latter case, these "typical" objects, called *medoids*, are used as seeds for the construction of groups in the technique presented in the next Section. Let us apply this idea to verify if the four groups obtained by a Ward clustering of the fish data are modified by a *k*-means partitioning.

*k-means with Ward species means per group (centroids) as starting points:*

```
# Mean species abundances on Ward site clusters
groups <- as.factor(spech.ward.g)
spe.means <- matrix(0, ncol(spe), length(levels(groups)))
row.names(spe.means) <- colnames(spe)
for (i in 1:ncol(spe)) {
  spe.means[i, ] <- tapply(spe.norm[, i], spech.ward.g, mean)
}
# Mean species abundances as starting points
startpoints <- t(spe.means)
# k-means on starting points
spe.kmeans2 <- kmeans(spe.norm, centers = startpoints)
```

*A slightly different approach is to go back to the hierarchical clustering, identify the most "typical" object in each group (cf. silhouette plot), and provide these medoids as starting points to* **`kmeans()`** *(argument* `centers`*):*

```
startobjects <- spe.norm[c(2, 17, 21, 23), ]
spe.kmeans3 <- kmeans(spe.norm, centers = startobjects)

# Comparison with the 4-group partition derived from
# Ward clustering:
table(spe.kmeans2$cluster, spech.ward.g)

# Comparison among the two optimized 4-group classifications:
table(spe.kmeans2$cluster, spe.kmeans3$cluster)

# Silhouette plot of the final partition
spech.ward.gk <- spe.kmeans2$cluster
k <- 4
sil <- silhouette(spech.ward.gk, spe.ch)
rownames(sil) <- row.names(spe)
plot(sil,
     main = "Silhouette plot - Ward & k-means",
     cex.names = 0.8,
     col = 2:(k + 1))
```

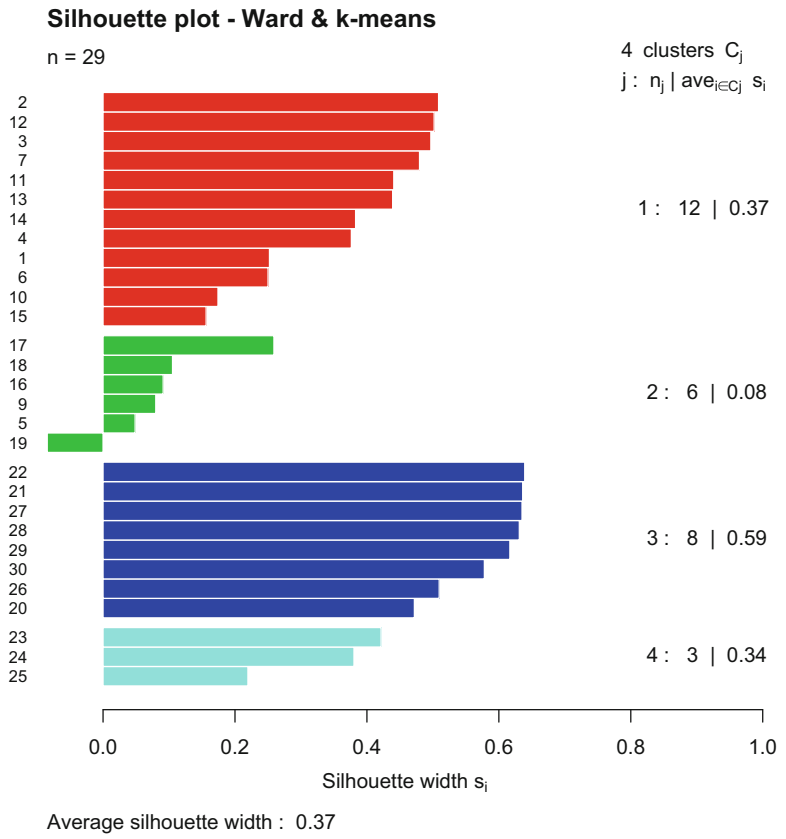


**Fig. 4.21** Silhouette plot of the four-group partition from Ward clustering optimized by *k*-means

As can be seen in the code, comparison between the original and the optimized groups, or between the two optimization results, can be made by means of contingency tables (see Sect. 4.9.2). The two optimizations (based on species means and on starting objects) produce the same result for our choice of 4 "typical" objects. Note that this choice is critical. Comparison with the original Ward 4-group result shows that only one object, #15, has been moved from cluster 2 to cluster 1. This improves the silhouette plot (Fig. 4.21) and changes slightly the map of the groups along the river (Fig. 4.22). Note that the membership of site #19 remains unclear: its silhouette value remains negative.

```
# Plot the optimized Ward clusters on a map of the Doubs River
drawmap(xy = spa,
        clusters = spech.ward.gk,
        main = "Four optimized Ward clusters along the Doubs River"
)
```

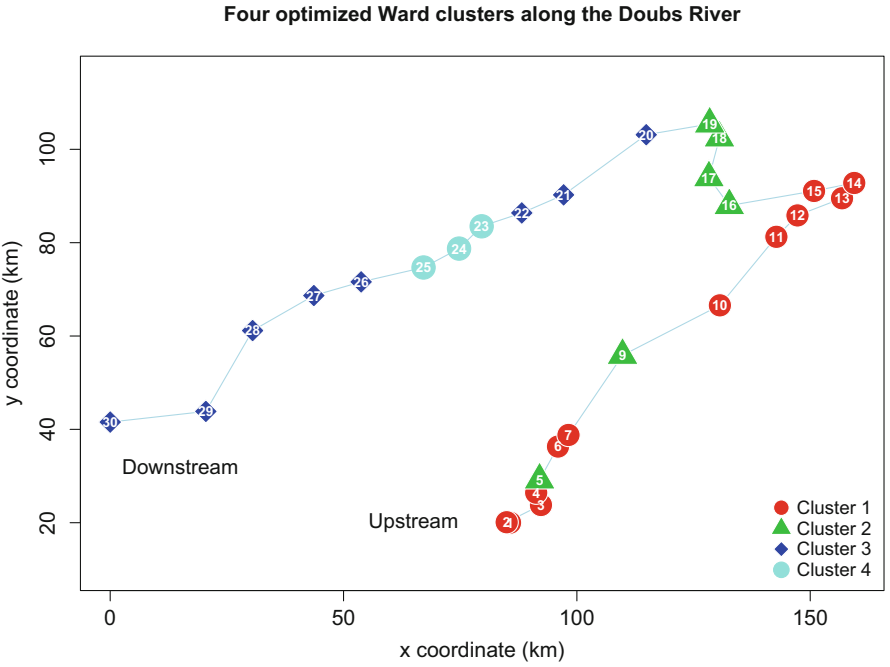


**Fig. 4.22** The four optimized Ward clusters on a map of the Doubs River

### *4.8.2 Partitioning Around Medoids (PAM)*

Partitioning around medoids (Chap. 2 *in* Kaufman and Rousseeuw 2005) *"searches for k representative objects or medoids among the observations of the dataset. These observations should represent the structure of the data. After finding a set of k medoids, k clusters are constructed by assigning each observation to the nearest medoid. The goal is to find k representative objects which minimize the sum of the dissimilarities of the observations to their closest representative object"* (excerpt from the **`pam()`** documentation file). By comparison, *k*-means minimizes the sum of the **squared** Euclidean distances within the groups. *k*-means is thus a traditional least-squares method, while PAM is not[2]. As implemented in **R**, **`pam()`** (package **`cluster`**) accepts raw data or dissimilarity matrices (an advantage over

[2]Many dissimilarity functions used in community ecology are non-Euclidean (for example the Jaccard, Sørensen and % difference indices). However, the square-root of these *D* functions is Euclidean. PAM minimizes the sum of the dissimilarities of the observations to their closest medoid. Since *D* is the square of the square-rooted (Euclidean) dissimilarities, PAM is a least-squares method when applied to these non-Euclidean *D* functions. See Legendre and De Cáceres (2013), Appendix S2.

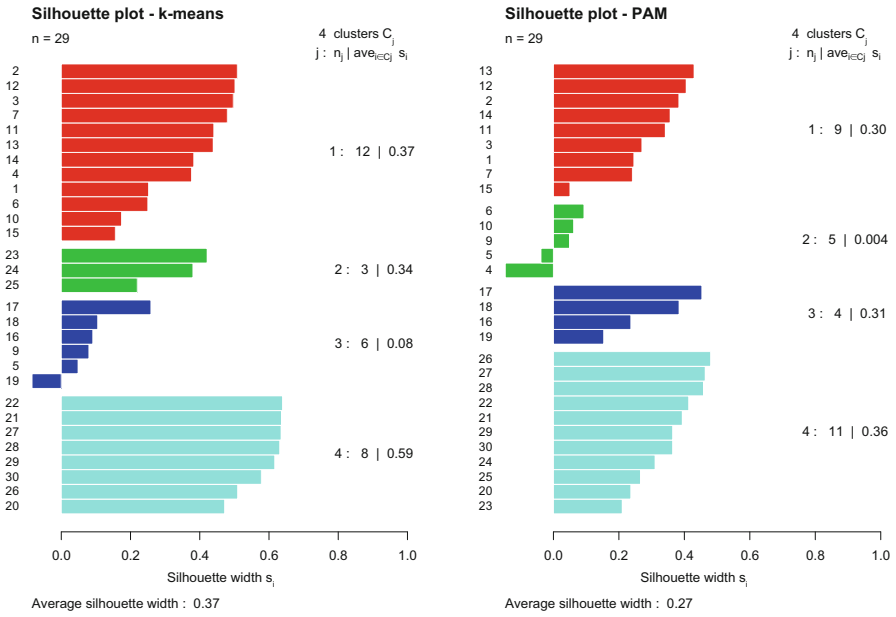


**Fig. 4.23** Silhouette plots of the *k*-means and PAM results

**`kmeans()`** since it broadens the choice of association measures) and allows the choice of an optimal number of groups using the silhouette criterion. The code below ends with a double silhouette plot comparing the *k*-means and PAM results (Fig. 4.23).

*Partitioning around medoids (PAM) computed from the chord distance matrix:*

```
# Choice of the number of clusters
# Loop: obtain average silhouette widths (asw) for 2 to 28 clusters
asw <- numeric(nrow(spe))
for (k in 2:(nrow(spe) - 1))
  asw[k] <- pam(spe.ch, k, diss = TRUE)$silinfo$avg.width
k.best <- which.max(asw)
plot(
  1:nrow(spe),
  asw,
  type = "h",
  main = "Choice of the number of clusters",
```

```
  xlab = "k (number of clusters)",
  ylab = "Average silhouette width"
)
axis(
  1,
  k.best,
  paste("optimum", k.best, sep = "\n"),
  col = "red",
  font = 2,
  col.axis = "red"
)
points(k.best,
       max(asw),
       pch = 16,
       col = "red",
       cex = 1.5)
```

*The not very interesting result k=2 is the best PAM solution with asw = 0.3841. Our previous choice of k=4 ends up with a poor performance in terms of silhouette width (asw = 0.2736). Nevertheless, let us compute a PAM for 4 groups:*

```
# PAM for k = 4 clusters
spe.ch.pam <- pam(spe.ch, k = 4, diss = TRUE)
summary(spe.ch.pam)

spe.ch.pam.g <- spe.ch.pam$clustering
spe.ch.pam$silinfo$widths

# Compare with classification from Ward clustering and from k-means
table(spe.ch.pam.g, spech.ward.g)
table(spe.ch.pam.g, spe.kmeans.g)
```

*The PAM result differs markedly from those of the Ward and k-means clusterings.*

```
# Silhouette profile for k = 4 groups, k-means and PAM
par(mfrow = c(1, 2))
k <- 4
sil <- silhouette(spe.kmeans.g, spe.ch)
rownames(sil) <- row.names(spe)
plot(sil,
     main = "Silhouette plot - k-means",
     cex.names = 0.8,
     col = 2:(k + 1))
plot(
  silhouette(spe.ch.pam),
  main = "Silhouette plot - PAM",
  cex.names = 0.8,
  col = 2:(k + 1)
)
```

*On the basis on this plot, you should be able to tell which solution (PAMor k-means) has a better silhouette profile.*
*You could also compare this result with the silhouette plot of the optimized Ward clustering produced earlier. Except cluster numbering, is there any difference between k-means and optimized Ward classifications? You can also address this question by examining a contingency table:*

```
# Compare classifications from k-means and optimized Ward
# clustering
table(spe.kmeans.g, spech.ward.gk)
```

*Hint* *The PAM method is presented as "robust" because it minimizes a sum of dissimilarities instead of a sum of squared Euclidean distances. It is also robust in that it tends to converge to the same solution with a wide array of starting medoids for a given k value; this does not guarantee, however, that the solution is the most appropriate for a given research purpose.*

This example shows that even two methods that are devoted to the same goal and belong to the same general class (here non-hierarchical clustering) may provide diverging results. It is up to the user to choose the one that yields classifications that are bringing out more pertinent information or are more closely interpretable using environmental variables (next section).

## 4.9 Comparison with Environmental Data

All methods above have been presented with examples on species abundance data. They can actually be applied to any other type of data as well, particularly environmental data tables. Of course, care must be taken with respect to the choice of the proper coding and transformation for each variable (Chap. 2) and of the association measure (Chap. 3).

### *4.9.1 Comparing a Typology with External Data (ANOVA Approach)*

We have seen that internal criteria, such as silhouette or other clustering quality indices, which rely on the species data only, were not always sufficient to select the "best" partition of the sites. The final choice of a typology should be based on the ecological interpretability of the groups. It could be seen as an external validation of the site typology.

Confronting clustering results (considered as response data) with external, independent explanatory data could be done by discriminant analysis (Sect. 6.5). From another point of view, the clusters obtained from the community composition data can be considered as a factor, or classification criterion, in the ANOVA sense. Here is a simplified example, showing how to perform quick assessments of the ANOVA assumptions (normality of residuals and homogeneity of variances) on several environmental variables separately, followed either by a parametric ANOVA or by a non-parametric Kruskal-Wallis test. Boxplots of the environmental variables (after some simple transformations to improve normality) for the four optimized Ward groups are also provided (Fig. 4.24). Note that, despite the fact that the clustering result based on the community composition data acts as an explanatory variable (factor) in the ANOVA, ecologically speaking we really look for an environmental interpretation of the groups of sites.

```
# Test of ANOVA assumptions
with(env, {
  # Normality of residuals
  shapiro.test(resid(aov(sqrt(ele) ~ as.factor(spech.ward.gk))))
  shapiro.test(resid(aov(log(slo) ~ as.factor(spech.ward.gk))))
  shapiro.test(resid(aov(oxy ~ as.factor(spech.ward.gk))))
  shapiro.test(resid(aov(sqrt(amm) ~ as.factor(spech.ward.gk))))
```

*Residuals of* `sqrt(ele)`, `log(slo)`, `oxy` *and* `sqrt(amm)` *are normally distributed, assuming that the power of the test is adequate. Try to find good normalizing transformations for the other variables.*

```
  # Homogeneity of variances
  bartlett.test(sqrt(ele), as.factor(spech.ward.gk))
  bartlett.test(log(slo), as.factor(spech.ward.gk))
  bartlett.test(oxy, as.factor(spech.ward.gk))
  bartlett.test(sqrt(amm), as.factor(spech.ward.gk))
```

*Variable* `sqrt(ele)` *has heterogeneous variances. It is not appropriate for parametric ANOVA.*

```
  # ANOVA of the testable variables
  summary(aov(log(slo) ~ as.factor(spech.ward.gk)))
  summary(aov(oxy ~ as.factor(spech.ward.gk)))
  summary(aov(sqrt(amm) ~ as.factor(spech.ward.gk)))

  # Kruskal-Wallis test of variable elevation
  kruskal.test(ele ~ as.factor(spech.ward.gk))
})
```

*Are slope, dissolved oxygen and dissolved ammonium significantly different among species clusters?*
*Does elevation differ among clusters?*

*Hints* *Note the use of* **`with()`** *at the beginning of the series of analyses to avoid the repetition of the name of the object* **`env`** *in each analysis. This is preferable to the use of* **`attach()`** *and* **`detach()`** *because the latter may lead to confusions if you have several datasets in your* **R** *console, and some happen to have variables with identical names.*

*The null hypothesis for the Shapiro test is that the variable is normally distributed; in the Bartlett test, $H_0$ states that the variances are equal among the groups. Therefore, for each of these tests, the p-value should be larger than the significance level, i.e. $P > 0.05$, for the ANOVA assumptions to be fulfilled.*

*The parametric Bartlett test is sensitive to departures from normality. For non-normal data, we provide a function called* **`bartlett.perm.R`** *that computes parametric, permutation and bootstrap (i.e., permutation with replacement) Bartlett tests.*

Two homemade generic functions will allow to perform post-hoc tests and show the results as letters on the boxplots of the environmental variables split by clusters (Fig. 4.24). Different letters denote significant differences among groups (in decreasing order). The **`boxplert()`** function should be used to perform ANOVA and LSD tests for multiple comparisons, whereas **`boxplerk()`** is used to perform a Kruskal-Wallis test and its corresponding post-hoc comparisons (both with Holm correction).

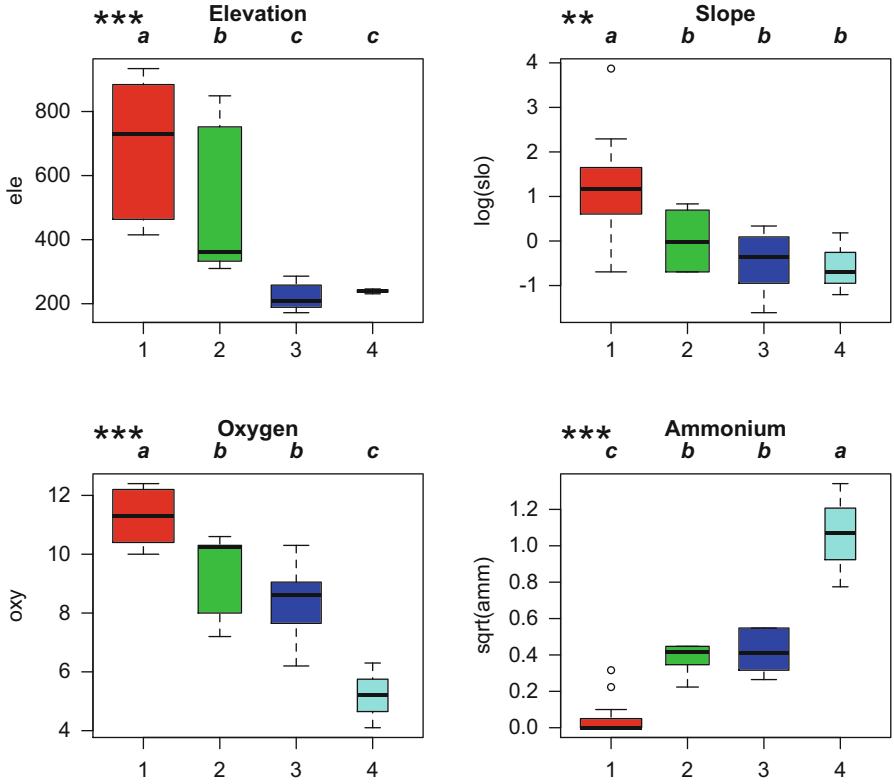


**Fig. 4.24** Boxplots of four environmental variables grouped according to the four species-based groups from the optimized Ward clustering. Stars indicate the significance of the differences among groups for each environmental variable. The letters indicate which group means are significantly different

```
# Use boxplert() or boxplerk() to plot results with post-hoc tests
with(env, {
  boxplerk(
    ele,
    spech.ward.gk,
    xlab = "",
    ylab = "ele",
    main = "Elevation",
    bcol = (1:k) + 1,
    p.adj = "holm"
  )
  boxplert(
    log(slo),
    spech.ward.gk,
    xlab = "",
    ylab = "log(slo)",
    main = "Slope",
    bcol = (1:k) + 1,
    p.adj = "holm"
  )
  boxplert(
    oxy,
    spech.ward.gk,
    xlab = "",
    ylab = "oxy",
    main = "Oxygen",
    bcol = (1:k) + 1,
    p.adj = "holm"
  )
  boxplert(
    sqrt(amm),
    spech.ward.gk,
    xlab = "",
    ylab = "sqrt(amm)",
    main = "Ammonium",
    bcol = (1:k) + 1,
    p.adj = "holm"
  )
})
```

> *How would you qualify the ecology of the four fish community types based on this analysis?*

Of course, the reverse procedure could be applied as well. One could cluster the environmental variables (to obtain a set of habitat types) and test if the species respond to these habitat types significantly through indicator species analysis (Sect. 4.11). In this approach, the species are tested one by one against the habitat types. Consider the question of multiple testing if several species are tested independently.

As an alternative, ordination-based multivariate approaches will be proposed in Chap. 6 to directly model and test the species-habitat relationships.

### 4.9.2 Comparing Two Typologies (Contingency Table Approach)

If you simply want to compare a typology generated from the species data to one independently obtained from the environmental variables, you can generate a table crossing the two typologies and test the relationship using a Fisher's exact test:

```
# Environment-based typology (see Chap. 2)
env2 <- env[, -1]
env.de <- vegdist(scale(env2), "euc")
env.kmeans <- kmeans(env.de, centers = 4, nstart = 100)
env.kmeans.g <- env.kmeans$cluster

# Table crossing the species and environment 4-group typologies
table(spe.kmeans.g, env.kmeans.g)
```

*Do the two typologies tell the same story?*

```
# Test the relationship using a Fisher's exact test
fisher.test(table(spe.kmeans.g, env.kmeans.g))
```

Such tables could also be generated using categorical explanatory variables, which can be directly compared with the species typology.

## 4.10 Species Assemblages

Many approaches exist to address the problem of identifying species associations in a data set. Here are some examples.

### 4.10.1 Simple Statistics on Group Contents

The preceding sections immediately suggest a way to define crude assemblages: compute simple statistics (for instance mean abundances) from typologies obtained

through a clustering method and look for species that are more present, abundant or specific in each cluster of sites. Here is an example.

```
# Compute mean species abundances in the four groups from the
# optimized Ward clustering
groups <- as.factor(spech.ward.gk)
spe.means <- matrix(0, ncol(spe), length(levels(groups)))
row.names(spe.means) <- colnames(spe)
for (i in 1:ncol(spe)) {
  spe.means[i, ] <- tapply(spe[, i], spech.ward.gk, mean)
}
group1 <- round(sort(spe.means[, 1], decreasing = TRUE), 2)
group2 <- round(sort(spe.means[, 2], decreasing = TRUE), 2)
group3 <- round(sort(spe.means[, 3], decreasing = TRUE), 2)
group4 <- round(sort(spe.means[, 4], decreasing = TRUE), 2)
# Species with abundances greater than group mean species abundance
group1.domin <- which(group1 > mean(group1))
group1
group1.domin
#... same for other groups
```

### 4.10.2 *Kendall's* W *Coefficient of Concordance*

Legendre (2005) proposed to use Kendall's *W* coefficient of concordance, together with permutation tests, to identify species assemblages in abundance data (this method cannot be applied to presence-absence data): *"An overall test of independence of all species is first carried out. If the null hypothesis is rejected, one looks for groups of correlated species and, within each group, tests the contribution of each species to the overall statistic, using a permutation test."* In this method, the search for species associations is done without any reference to a typology of the sites known *a priori* or computed from other data, for example environmental. The method aims at finding the most encompassing assemblages, i.e., the smallest number of groups containing the largest number of positively and significantly associated species.

The package **`kendall.W`** has been written to carry out these computations. Its functions are now part of **`vegan`**. The simulation results accompanying the Legendre (2005) paper show that *"when the number of judges [= species] is small, which is the case in most real-life applications of Kendall's test of concordance, the classical $\chi^2$ test is overly conservative, whereas the permutation test has correct Type I error; power of the permutation test is thus also higher."* The **`kendall.global()`** function also includes a parametric *F*-test which does not suffer from the problems of the $\chi^2$ test and has correct Type I error (Legendre 2010).

As a simple example, let us classify the fish species into several groups using *k*-means partitioning (Fig. 4.25), and run a global test (**`kendall.global()`**) to

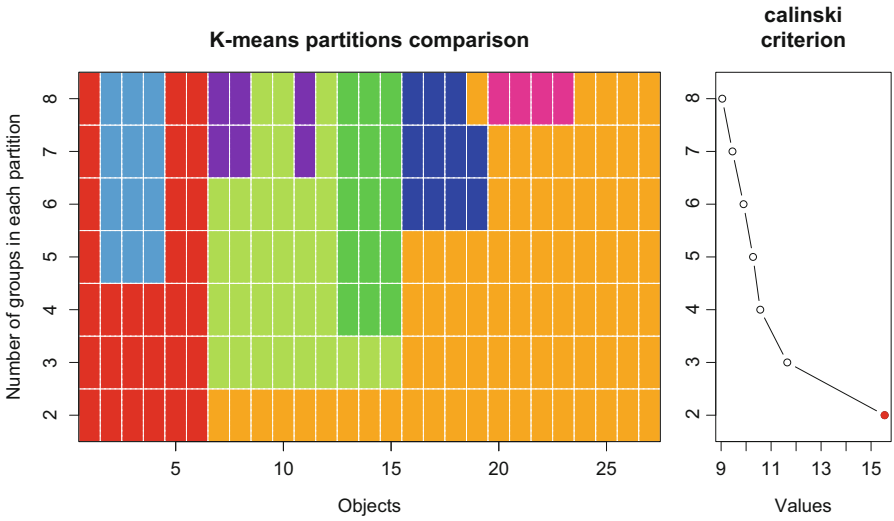


**Fig. 4.25** *k*-means cascade plot showing the R-mode partitioning of the fish species. The Calinski-Harabasz criterion points to an optimum of 2 groups

know if all groups of species (called "judges" in the original paper) are globally significantly associated. If it is the case, we shall run post hoc tests (**`kendall.post()`**) on the species of each group to verify if all species within a group are concordant[3].

```
# Transformation of species data and transposition
spe.hel <- decostand(spe, "hellinger")
spe.std <- decostand(spe.hel, "standardize")
spe.t <- t(spe.std)
```

*We can run a first test of Kendall concordance involving all species:*

```
(spe.kendall.global1 <- kendall.global(spe.hel))
```

[3]Technical note: in the code that follows, the species data are first Hellinger-transformed (see Sect. 3.5). Then they are standardized. Standardization, which makes the variables dimensionless, is necessary because Kendall's *W* is based on correlation coefficients. For coherence reasons, a clustering of the species data, before Kendall's concordance analysis, must also be computed in a space where the variables are dimensionless.

> *The null hypothesis of absence of concordance is rejected. Thus, we can look for groups of species, then proceed with the Kendall analysis of these groups*

```
# k-means partitioning of species
spe.t.kmeans.casc <- cascadeKM(
  spe.t,
  inf.gr = 2,
  sup.gr = 8,
  iter = 100,
  criterion = "calinski"
)
plot(spe.t.kmeans.casc, sortg = TRUE)
```

> *This result indicates that two groups may be a good choice. Avoid solutions with groups containing a single species except when it is clear that this species belongs to no other group. One group has 6 species and the other has 21. Three or four groups would also be fine at this point of the analysis: all groups would have 3 species or more.*

```
# The partition into 2 groups is found in column 1 of the
# object $partition
(clusters2 <- spe.t.kmeans.casc$partition[, 1])
```

Partitions into three or four groups:

```
(clusters3 <- spe.t.kmeans.casc$partition[, 2])
(clusters4 <- spe.t.kmeans.casc$partition[, 3])
```

We will now examine the division of the species in two groups. Let us run a global Kendall W test on each group. This is done with a single call to the **`kendall.global()`** function.

```
# Concordance analysis
(spe.kendall.global2 <- kendall.global(spe.hel, clusters2))
```

Look at the corrected permutational p-values. If all values are equal to or smaller than 0.05, you can consider that all groups are globally significant, i.e. that on the whole they contain species that are concordant; this does not mean that all species in a globally significant group are concordant, only that at least some species are. If the corrected p-values for some groups were not significant (it is not the case with this example), it would indicate that these groups include non-concordant species and should be subdivided into smaller groups. In other words, a partition into more than 2 groups would be in order.

Now let us run *a posteriori* tests to identify the significantly concordant species within each group:

```
# A posteriori tests
(spe.kendall.post2 <- kendall.post(spe.hel, clusters2,
                                   nperm = 9999))
```

Look at the mean Spearman correlation coefficients of the individual species. A group contains concordant species if each of its species has a positive mean correlation with all the other species of its group. If a species has a negative mean correlation with all other members of its group, this indicates that this species should be left out of the group. Try a finer division of the groups and see if that species finds itself in a group for which it has a positive mean correlation. This species may also form a singleton, i.e. a group with a single species.

With 2 groups, we have in the largest group one species (`Sqce`) that has a negative mean correlation with all members of its group. This indicates that we should look for a finer partition of the species. Let us carry out a posteriori tests with 3 groups. Readers can also try with 4 groups and examine the results.

```
(spe.kendall.post3 <- kendall.post(spe.hel, clusters3,
                                   nperm = 9999))
```

Now all species in the three groups have positive mean Spearman correlations with the other members of their group. `Sqce` finds itself in a new group of 9 species with which it has a mean positive correlation, although its contribution to the concordance of that group is not significant. All the other species in all three groups contribute significantly to the concordance of their group. So we can stop the analysis and consider that three groups of species, with respectively 12, 9 and 6 species, adequately describe the species associations in the Doubs River.

Ecological theory predicts nested structures in ecological relationships. Within communities, subgroups of species can be more or less loosely or densely associated. One can explore such avenues by investigating smaller species groups within the large species associations revealed by the Kendall *W* test results.

The groups of species defined here may be further interpreted ecologically by different means. For instance, mapping their abundances along the river and computing summary statistics on the sites occupied by the species assemblages can help in assessing their ecological roles. Another avenue towards interpretation is to compute a redundancy analysis (RDA, Sect. 6.3) of the significantly associated species with respect to a set of explanatory environmental variables.

### *4.10.3 Species Assemblages in Presence-Absence Data*

A method exists for presence-absence data (Clua et al. 2010). It consists in computing the *a* component of Jaccard's $S_7$ coefficient (as a measure of co-occurrence among species) in R-mode and assessing its probability by means of a permutation

test in the spirit of the Raup and Crick (1979) coefficient. The p-values act as dissimilarities: they have very small values for highly co-occurring species. We provide an **R** function called **`test.a()`** to compute this coefficient. Readers are invited to apply it to the species of the Doubs fish data transformed to presence-absence form. A critical point is to specify enough permutations for the probabilities to survive a correction for multiple testing (see Sect. 7.2.6). There are 27 species, and thus $27 \times 26/2 = 351$ tests will be run. A Bonferroni correction requires a p-value of $0.05/351 = 0.0001425$ to remain significant at the 0.05 level. This requires at least 9999 permutations, since the smallest p-value would be $1/(9999 + 1) = 0.0001$. 99,999 permutations provide a finer estimation of the p-value but beware: computation can take several minutes.

```
# Transform the data to presence-absence
spe.pa <- decostand(spe, "pa")
# Test the co-occurrence of species
res <- test.a(spe.pa, nperm = 99999)
summary(res)
```

*In the output object,* `res$p.a.dist` *contains a matrix of p-values of class dist. The next step is to compute a Holm correction (see Sect. 7.2.6) on the matrix of p-values unfolded as a vector.*

```
# Compute a Holm correction on the matrix of p-values
res.p.vec <- as.vector(res$p.a.dist)
adjust.res <- p.adjust(res.p.vec, method = "holm")
range(adjust.res)
```

*Among the corrected Holm p-values, find 0.05 or the closest value smaller than 0.05:*

```
(adj.sigth <- max(adjust.res[adjust.res <= 0.05]))
```

*Now find the uncorrected p-value corresponding to* `adj.sigth`*:*

```
(sigth <- max(res.p.vec[adjust.res <= 0.05]))
```

*In this run it is 0.00017. The significant values thus have an uncorrected probability of 0.00017 or less. Replace all larger values in the probability matrix by 1:*

```
res.pa.dist <- res$p.a.dist
res.pa.dist[res.pa.dist > sigth] <- 1
```

*How many (unadjusted) values are equal to or smaller than* `sigth`*?*

```
length(which(res.p.vec <= sigth))
```

*The dissimilarity matrix made of the p-values can be displayed as a heat map (see Chap. 3):*

```
# Heat map of significant p-values
coldiss(res.pa.dist,
        nc = 10,
        byrank = TRUE,
        diag = TRUE)
```

### 4.10.4 Species Co-occurrence Network

Co-occurrence network analysis becomes more and more popular in community ecology, especially to investigate ecological interactions among species or among communities. The principle is to analyse associations among species based on their co-occurrence in ecological meta-communities or multitrophic species assemblages. Based on the topology of the co-occurrence network, sociological groups of species are defined, called "modules". Network structure is characterized by two main properties, namely "modularity" (the extent to which species co-occurrences are organized into modules, i.e. densely connected, non-overlapping subsets of species) and "nestedness" (the tendency of the network to show a nested pattern wherein the species composition of small assemblages is a nested subset of larger assemblages; see Sect. 8.4.3). The role of a species is defined by its position compared with other species in its own module (its "standardized within-module degree", i.e., the number of links that a node has with other nodes in the same module, standardized by the mean and standard deviation of the number of links per node in the module) and how well it connects to species in other modules (its among-module "connectivity"). For more details, see e.g. Olesen et al. (2007) and Borthagaray et al. (2014).

Basically, the network is built from an adjacency matrix, which may be binary (species significantly co-occur or not) or numeric (links among species are weighted). Various metrics of positive, neutral or negative association among

species can be used to compute the adjacency matrix, including Spearman correlations, Jaccard similarity coefficients or the p-values obtained in the previous section. These metrics can be computed from a species presence-absence or abundance matrix. A threshold is often applied to restrict the non-zero links to the most important positive associations to build undirected networks. The network is drawn in such a way that frequently co-occurring species are located close to one another in the graph, using various algorithms.

Several **R** packages are devoted to network analysis. Restricting ourselves to a basic introduction to this approach, we shall use **`igraph`** for network processing and **`picante`** for computing co-occurrence distances (Hardy 2008).

Let us begin by computing several adjacency matrices from the fish species dataset used above. You can choose one of these symmetric matrices to build the undirected co-occurrence network, here from Jaccard dissimilarity (Fig. 4.26).

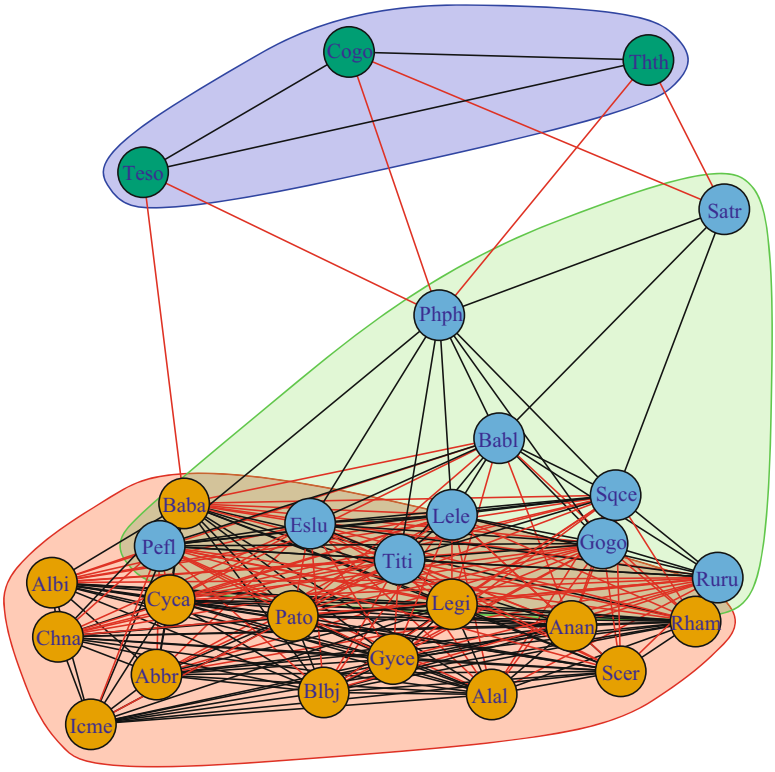


**Fig. 4.26** Graph of the co-occurrence network of the fish species based on Jaccard similarity, showing three modules (species bubble colours). Positive intra-module associations are indicated by black lines, positive inter-module ones by red lines

```
# Attach supplementary packages
library(igraph)
library(rgexf)

# Adjacency matrix from the binary matrix of significant
# co-occurrences (results of the previous section)
adjm1 <- 1 - as.matrix(res.pa.dist)
diag(adjm1) <- 0

# Adjacency matrix from the "a" distance matrix
adjm2 <- 1 - as.matrix(res$p.a.dist)
adjm2[adjm2 < 0.5] <- 0
diag(adjm2) <- 0

# Adjacency matrix from the Spearman rank correlation matrix
adjm3 <- cor(spe, method = "spearman")
# Only positive associations (rho >= 0.25)
adjm2[adjm3 < 0.25] <- 0
adjm2[adjm3 >= 0.25] <- 1  # binary co-occurrences
diag(adjm3) <- 0

# Species co-occurrence dissimilarities
# (picante package, Hardy 2008)
adjm4 <- species.dist(spe.pa, metric = "jaccard")
adjm4 <- as.matrix(adjm4)
adjm4[adjm4 < 0.4] <- 0

# Select an adjacency matrix
adjm <- adjm4
summary(as.vector(adjm))


# Plot histogram of adjacency values
hist(adjm)

# Build graph
go <- graph_from_adjacency_matrix(adjm, weighted = TRUE,
                                  mode = "undirected")
plot(go)

# Network structure detection: find densely connected subgraphs
# (modules) in a graph
wc <- cluster_optimal(go)
modularity(wc)
membership(wc)
plot(wc, go)

# Detach package rgexf
detach("package:rgexf", unload = TRUE)
# If not sufficient:
unloadnamespace("rgexf")
```

## 4.11 Indicator Species

### *4.11.1 Introduction*

The search for indicator or characteristic species in groups of sites is a particularly important question in applied ecology, where conservation and management specialists look for methods to quickly assess or predict the type of environment characterizing a given location, to assess the type of living community (e.g., vegetation type) in a given area, or to monitor temporal changes in some ecological variables (e.g., mean annual temperature). In fundamental ecology, one often looks for ecological preferences of species among a set of site groups, either to gain knowledge about the species themselves or to improve the knowledge of the habitats sampled in the groups of sites. These approaches are complementary, and different methods exist to address them. De Cáceres and Legendre (2009) presented the two complementary approaches and their related indices, classified into indicator value and correlation indices, respectively.

**Indicator value indices** are based on the concepts of *specificity* (highest when the species is present in the target group but not elsewhere) and *fidelity* (highest when the species is present in all sites of the target group). A high indicator value is obtained by a combination of high specificity and fidelity.

**Correlation indices** rely on the computation of statistics related to Pearson's *r* correlation coefficient, measuring the relationship between a vector indicating the presence or abundance of a species at different sites and a vector describing to which predefined group each site belongs.

De Cáceres and Legendre (2009) describe 12 indices belonging to the two groups. They state that indicator value indices are more useful for assessing the species predictive values as bioindicators, e.g. for field determination of community types or for ecological monitoring, whereas correlation indices should be used to determine the ecological preference of a given species among a set of alternative site groups (De Cáceres and Legendre 2009, p. 3573).

### *4.11.2 IndVal: Species Indicator Values*

The Dufrêne and Legendre (1997) IndVal index is computed as the product of the specificity of a species to the targeted group by its fidelity to the targeted group. Specificity is defined by the mean abundance of the species within the targeted group compared to its mean abundance across the groups; fidelity is the proportion of sites of the targeted group where the species is present. Dave Roberts, the author of the package **`labdsv`** used hereunder, summarized the concept as follows (pers. comm.): "*The indval approach looks for species that are both necessary and sufficient, i.e. if you find that species you should be in that type, and if you are in that type you should find that species*".

The groups of sites can be defined in various ways. A simple, although circular way is to use the result of a cluster analysis based on the species data. The indicator species are then simply the most prominent members of these groups. Of course, one cannot rely on statistical tests to identify the indicator species in this case, since the classification and the indicator species do not derive from independent data. Another approach, which is conceptually and statistically more fruitful, consists in clustering the sites on the basis of independent data (environmental variables, for instance). The indicator species can then be considered *indicator* in the true sense of the word, i.e. species closely related to the ecological conditions of their group. The statistical significance of the indicator values (i.e., the probability of obtaining by chance as high an indicator value as observed) is assessed by means of a permutation test.

The Dufrêne and Legendre index is available in function **`indval()`** of package **`labdsv`**. Let us apply it to our fish data. As an example of a search for indicator species related to one explanatory variable, we could for instance divide the data set into groups of contiguous sites on the basis of the variable `dfs` (distance from the source), acting as a surrogate of the overall river gradient conditions, and look for indicator species in the groups.

```
# Divide the sites into 4 groups depending on the distance from
# the source of the river
dfs.D1 <- dist(data.frame(dfs = env[, 1],
               row.names = rownames(env)))
dfsD1.kmeans <- kmeans(dfs.D1, centers = 4, nstart = 100)
# Cluster delimitation and numbering
dfsD1.kmeans$cluster
# Cluster labels are in arbitrary order
# See Hint below
grps <- rep(1:4, c(8, 10, 6, 5))
# Indicator species for this typology of the sites
(iva <- indval(spe, grps, numitr = 10000))
```

*Hint* ***BEWARE****: in the output table, the cluster numbering corresponds to the cluster numbers fed to the program, which do not necessarily follow a meaningful order. The k-means analysis produces arbitrary group labels in the form of numbers. The column headers of the* **`indval()`** *output are the ones provided to the program, but the columns have been reordered according to the labels. Consequently, the groups do not follow the order along the river. This is why we have manually constructed an object (`grps`) with sequential group numbers.*

The resulting object contains the following tables:

- `relfrq` = relative frequency of the species in each group = number of sites where species is present/number of sites in the group
- `relabu` = relative abundance of the species across groups = total abundance in group/grand total abundance
- `indval` = indicator value (IndVal) of each species
- `maxcls` = cluster where the species has highest IndVal
- `indcls` = highest IndVal of the species
- `pval` = permutational p-value of IndVal

Correct the p-values for multiple testing:

```
pval.adj <- p.adjust(iva$pval)
```

The following lines of code extract the significant indicator species with their group with highest IndVal, the corresponding IndVal, the species' indicator p-value and their total frequency in the data set.

```
# Table of the significant indicator species
gr <- iva$maxcls[pval.adj <= 0.05]
iv <- iva$indcls[pval.adj <= 0.05]
pv <- iva$pval[pval.adj <= 0.05]
fr <- apply(spe > 0, 2, sum)[pval.adj <= 0.05]
fidg <- data.frame(
  group = gr,
  indval = iv,
  pvalue = pv,
  freq = fr
)
fidg <- fidg[order(fidg$group, -fidg$indval), ]
fidg

# Export the result to a CSV file (to be opened in a spreadsheet)
write.csv(fidg, "IndVal-dfs.csv")
```

> *On this basis, what do you think of the results? How do they relate to the four ecological zones presented in Chap.1?*
> *Note that the indicator species identified here may differ from the members of the species assemblages identified in Sect. 4.10. The indicator species are linked to predefined groups, whereas the species assemblages are identified without any prior classification or reference to environmental conditions.*

In Sect. 4.12 you will find another application in relationship with the MRT method. Some detailed results will be presented.

Package **indicspecies**, which was produced as a companion package to the De Cáceres and Legendre (2009) paper, computes various indicator species indices, including IndVal (or, actually, the square root of IndVal). Two especially interesting features of this package are the possibility of pooling two or more groups of a typology in turn to look for species that may be indicators of pooled groups (in function **multipatt()**) and the computation of bootstrapped confidence intervals around indicator values (argument `nboot` of function **strassoc()**).

Let us first compute the same analysis as above, using function **multipatt()** this time and taking advantage of the possibility of pooling the groups by two.

```
# Indval with indicspecies::multipatt() with search for indicator
# species of pooled groups
(iva2 <- multipatt(
  spe,
  grps,
  max.order = 2,
  control = how(nperm = 999)
))
```

In function **multipatt()**, the default indicator measure is `func = IndVal.g`. The ".g" indicates that the measure is corrected for unequal group sizes. This corresponds to the original Dufrêne and Legendre (1997) IndVal measure and is recommended.

The output object contains several matrices:

- `comb` = matrix of the belonging of the sites to all possible combinations up to the order requested in the analysis;
- `str` = the "strength of association", i.e., the value of the chosen index;
- `A` = if the function is the IndVal index, value of its A component (specificity); otherwise NULL;
- `B` = if the function is the IndVal index, value of its B component (fidelity); otherwise NULL;
- `sign` = results of the best patterns, i.e., the group or combination of groups where the statistic is the highest, and permutation test results for that result. No correction for multiple testing is provided.

Here again, the p-values should be corrected for multiple testing:

```
(pval.adj2 <- p.adjust(iva2$sign$p.value))
```

One can also request a simpler output with the **summary()** function:

```
summary(iva2, indvalcomp = TRUE)
```

The summary displays the groups and combinations of groups (if any has been requested) with significant indicator species. In each of these groups it displays the species' IndVal value (stat) and the p-value of the permutational test. With argument

`indvalcomp = TRUE`, the A (specificity) and B (fidelity) components of the IndVal index are displayed as well.

The results of this second analysis are slightly different from the previous ones because for some species the highest IndVal is found for a combination of groups instead of a single group. For instance, for the brown trout (`Satr`) the highest value is found for the combination of groups 1 + 2 whereas, if the analysis is conducted on separate groups only, the highest IndVal is in group 1.

Our next analysis consists in computing confidence intervals around IndVal values using function **`strassoc()`**. The number of iterations is provided by argument `nboot`.

```
# Indval with bootstrap confidence intervals of indicator values
(iva2.boot <- strassoc(spe, grps, func = "IndVal.g", nboot = 1000))
```

The output object consists in a list of three elements: the indicator value (`$stat`), and the lower (`$lowerCI`) and upper (`$upperCI`) limits of the confidence intervals. For the first three species, the results are the following (they vary from run to run since they result from bootstrapping):

```
$stat
            1           2           3           4
Cogo 0.0000000 0.89442719 0.00000000 0.00000000
Satr 0.6437963 0.67667920 0.00000000 0.05923489
Phph 0.5784106 0.75731725 0.09901475 0.05423261
(...)
$lowerCI
            1           2           3           4
Cogo 0.00000000 0.70710678 0.00000000 0.00000000
Satr 0.39086798 0.45825757 0.00000000 0.00000000
Phph 0.30618622 0.58177447 0.00000000 0.00000000
(...)
$upperCI
            1           2           3           4
Cogo 0.0000000 1.0000000 0.0000000 0.0000000
Satr 0.8277591 0.8537058 0.0000000 0.1924501
Phph 0.7585133 0.9004503 0.2348881 0.1721326
(...)
```

When interpreting confidence intervals, keep in mind that any indicator value whose lower limit is equal to 0 can be considered non-significant, even if the value itself is greater than 0. For instance, in groups 3 and 4, the IndVal values of the Eurasian minnow (`Phph`) are 0.09 and 0.054 respectively, but the lower limit of the confidence intervals is 0. Also, two values whose confidence intervals are overlapping cannot be considered different. For example, in group 1, the CI limits for the brown trout (`Satr`) and the Eurasian minnow (`Phph`) are [0.391; 0.828] and

[0.306; 0.758]. Therefore, there is a large probability that the true IndVal values for these two species reside somewhere within the common part of these ranges, i.e., [0.391; 0.758], but there is no way of determining that one value is larger than the other.

### *4.11.3 Correlation-Type Indices*

As mentioned above, correlation indices were devised to help identify the ecological preferences of species among a set of groups. De Cáceres and Legendre (2009) pointed out that this approach is "*probably more useful* [for this purpose] *than the indicator value approach, because the former naturally allows the detection of negative preferences*". Beware, however, of the problems relative to the absence of species from a set of sites. The absence of a species may be due to different reasons in each site, and thus it should not be interpreted in ecological terms without due caution (see Sect. 3.2.2). Species with high ecological preferences for a given group of sites representing well-defined ecological conditions are often called "diagnostic" species in plant ecology and are useful to identify vegetation types in field surveys (Chytrý et al. 2002 *in* De Cáceres and Legendre 2009).

The simplest correlation-type index for presence-absence data is called Pearson's $\phi$ (phi) coefficient of association (Chytrý et al. 2002 *in* De Cáceres and Legendre 2009). It consists in the correlation between two binary vectors. For all sites, one vector gives the presence or absence of the species, and the other indicates that the site belongs (1) or not (0) to a given group. With quantitative (abundance) data, the phi coefficient is called the point biserial correlation coefficient and a vector of abundances replaces the presence-absence vector. In function **`strassoc()`**, the default indicator measure is the phi coefficient, requested by `func = "r"`, but an `"r.g"` option is available that corrects the measure for unequal group sizes. We recommend the latter option, following De Cáceres' **`indicspecies`** tutorial.

Let us compute the phi coefficient (corrected for unequal group sizes) on the fish abundance data.

```
# Phi correlation index
iva.phi <- multipatt(
  spe,
  grps,
  func = "r.g",
  max.order = 2,
  control = how(nperm = 999)
)
summary(iva.phi)
```

The results of this analysis look quite similar to that of the IndVal analysis. Note that the statistical tests highlight the highest *positive* phi values (see the summary). Therefore, it is natural that both approaches (species indicator values and

correlation) highlight the same strongest associations between species and selected groups. However, it may be interesting to display all the phi values to identify possible sets of environmental conditions (represented by the groups, if these have been defined on the basis of environmental variables) that are *avoided* by some species (with the caveat mentioned above):

```
round(iva.phi$str, 3)
```

For instance, the bleak (`Alal`) shows a very strong negative association ($\phi = -0.92$) with the group of sites "1 + 2", which mirrors its strongest positive association ($\phi = 0.92$) with the group of sites "3 + 4". Indeed, the bleak is absent from all but 3 sites of groups 1 + 2, and is present in all sites of groups 3 + 4. Obviously, the bleak prefers the lower regions of the river and avoids the higher ones. The ecological characteristics of the various sections of the Doubs River are described and analysed in various sections of this book.

It is also possible to compute bootstrap confidence intervals around the phi values, using the same **`strassoc()`** function as was used to compute the IndVal coefficient:

```
iva.phi.boot <- strassoc(spe, grps, func = "r.g", nboot = 1000)
```

Since the phi values can be negative or positive, you can use this opportunity to obtain a bootstrap test of significance of the phi values: all values where both confidence limits have the same sign (i.e., CI not encompassing 0) are deemed significant. This complements the permutational tests for the negative values. For instance, the avoidance of the bullhead (`Cogo`) of the conditions prevailing in group 1 is significant in that sense, the CI interval being $[-0.313, -0.209]$.

## 4.12 Multivariate Regression Trees (MRT): Constrained Clustering

### *4.12.1 Introduction*

Multivariate regression trees (MRT; De'ath 2002) are an extension of univariate regression trees, a method allowing the recursive partitioning of a quantitative response variable under the control of a set of quantitative or categorical explanatory variables (Breiman et al. 1984). Such a procedure is sometimes called constrained or supervised clustering. The result is a tree whose "leaves" (terminal groups of sites) are composed of subsets of sites chosen to minimize the within-group sums of squares (as in a *k*-means clustering), but where each successive partition is defined by a threshold value or a state of one of the explanatory variables. Among the numerous potential solutions in terms of group composition and number of leaves, one usually retains the one that has the best *predictive* power. This stresses the fact

that, contrary to most constrained ordination methods described in Chap. 6, where the selection of explanatory variables is made on the basis of *explanatory* power, MRT focuses on prediction, making it a very interesting tool for practical applications, as in environmental management. The focus on prediction is embedded in the method, as will become clear below.

MRT is a powerful and robust method, which can handle a wide variety of situations, even those where some values are missing, and where the relationships between the response and explanatory variables are nonlinear, or where high-order interactions among explanatory variables are present.

## *4.12.2 Computation (Principle)*

The computation of a MRT consists in two procedures running together: (1) constrained partitioning of the data and (2) cross-validation of the results. Let us first briefly explain the two procedures. After that we will see how they are applied together to produce a model that has the form of a decision tree.

### 4.12.2.1 Constrained Partitioning of the Data

- For each explanatory variable, produce all possible partitions of the sites into two groups. For a quantitative variable, this is done by sorting the sites according to the ordered values of the variable, and repeatedly splitting the series after the first, second... $(n-1)^{\text{th}}$ object. For a categorical variable, allocate the objects to two groups, screening all possible combinations of levels. In all cases, compute the resulting sum of within-group sums of squared distances to the group means (within-group SS) for the response data. Retain the solution minimizing this quantity, along with the identity and value of the explanatory variable or the level of the categorical variable producing the partition retained.
- Repeat the same procedure within each of the two subgroups retained above; in each group, retain the best partition along with the corresponding explanatory variable and its threshold value.
- Continue within all partitions until all objects form their own group or until a preselected smallest number of objects per group is reached. At that point, select the tree with size (number of groups) appropriate to the aims of the study. For studies with a predictive objective, cross-validation, which is a procedure to identify the best predictive tree, is developed below.
- Apart from the number and composition of the leaves, an important characteristic of a tree is its *relative error* (RE), i.e., the sum of the within-group SS over all leaves divided by the overall SS of the data. In other words, this is the fraction of variance not explained by the tree. Without cross-validation, among the successive partitioning levels, one would retain the solution minimizing RE; this would

be equivalent to retaining the solution maximizing the $R^2$. However, this would be an explanatory rather than predictive approach. De'ath (2002) states that "*RE gives an over-optimistic estimate of how accurately a tree will predict for new data, and predictive accuracy is better estimated from the cross-validated relative error (CVRE)*".

#### 4.12.2.2 Cross-Validation of the Partitions and Pruning of the Tree

At which level should we prune a tree, i.e., cut each of its branches so as to retain the most sensible partition? To answer this question in a prediction-oriented manner, one uses a subset of the objects (training set) to construct the tree, and the remaining objects (test set) to validate the result by allocating them to the constructed

The measure of predictive error is the cross-validated relative error (CVRE). The function is:

$$CVRE = \frac{\sum_{k=1}^{\nu} \sum_{i=1}^{n} \sum_{j=1}^{p} \left( y_{ij(k)} - \widehat{y}_{j(k)} \right)^2}{\sum_{i=1}^{n} \sum_{j=1}^{p} \left( y_{ij} - \bar{y}_j \right)^2} \tag{4.1}$$

where $y_{ij(k)}$ is one observation of the test set $k$, $\widehat{y}_{j(k)}$ is the predicted value of one observation in one leaf (centroid of the sites of that leaf), and the denominator represents the overall dispersion (sum of squares) of the response data.

The CVRE can thus be defined as the ratio between the dispersion unexplained by the tree (summed over the $k$ test sets) divided by the overall dispersion of the response data. Of course, the numerator changes after every partitioning event. CVRE is 0 for perfect predictors and close to 1 for a poor set of predictors.

#### 4.12.2.3 MRT Procedure

Now that we have both components of the methods, let us put them together to explain the sequence of events of a cross-validated MRT run:

- Randomly split the data into $k$ groups; by default $k = 10$.
- Leave one of the $k$ groups out and build a tree by constrained partitioning, the decisions being made on the basis of the minimal within-group SS.
- Rerun the step above $k - 1$ times, leaving out each of the test groups in turn.
- In each of the $k$ solutions above, and for each possible partition size (number of groups) within these solutions, reallocate the test set. Compute the CVRE for all partition sizes of the $k$ solutions (one CVRE value per size). Eq. 4.1 encompasses the computation for one level of partitioning and all $k$ solutions.
- Pruning of the tree: retain the partition size for which the CVRE is smallest. An alternative solution is to retain a smallest size for which the CVRE value is the

minimal CVRE value plus one standard error of the CVRE values. This is called "the 1 SE rule".
- To obtain an estimate of the error of this process, run it a large number of times (100 or 500 times) with other random assignments of the objects to *k* groups.
- The final tree retained is the one showing most of the smallest CVRE values over all permutations, or the one respecting most often the 1 SE rule.
- In **mvpart**, the smallest possible number of objects in one group is equal to `ceiling(log2(n))` when argument `minauto = TRUE`. Setting `minauto = FALSE` allows the partitioning to proceed until the data set is fully split into individual observations.

In MRT analysis, the computations of sums-of-squares (SS) are done in Euclidean space. To account for special characteristics of the data, they can be pre-transformed prior to being submitted to the procedure. Pre-transformations for species data prior to their analysis by Euclidean-based methods are presented in Sect. 2.2.4. For response data that are environmental descriptors with different physical units, the variables should be standardized before they are used in MRT.

### 4.12.3 *Application Using Packages* **mvpart** *and* **MVPARTwrap**

As of this writing, the only package implementing a complete and handy version of MRT is **mvpart**. Unfortunately, this package is no longer supported by the R Core Team, so that no updates are available for **R** versions posterior to R 3.0.3. Nevertheless, **mvpart** can still be installed on more recent versions of **R** by applying the following code.

```
# On Windows machines, Rtools (3.4 and above) must be installed
# first. Go to: https://cran.r-project.org/bin/windows/Rtools/
# After that (for Windows and MacOS), type:
install.packages("devtools")
library(devtools)
install_github("cran/mvpart", force = TRUE)
install_github("cran/MVPARTwrap", force = TRUE)
```

Package **mvpart** has been written to compute MRT, using univariate regression trees computed by a function called **rpart()**. Its use requires that the response data belong to class 'matrix' and the explanatory variables to class 'data frame'. The relationship is written as a formula of the same type as those used in regression functions (see **?lm**). The example below shows the simplest implementation, where one uses all the variables contained in the explanatory data frame.

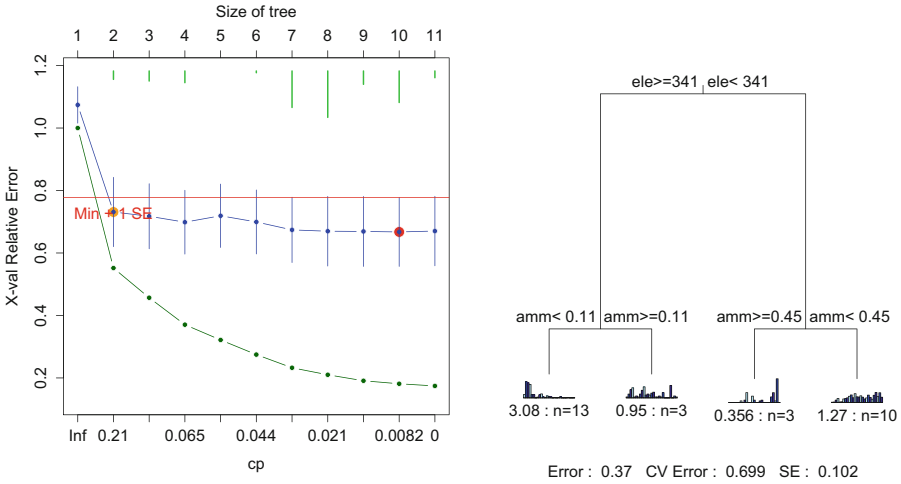


**Fig. 4.27** Left: graph of the (steadily decreasing) relative error RE and the cross-validated relative error CVRE. The solution with the smallest CVRE is indicated (red point), as well as CVRE error bars. The green vertical bars indicate the number of times that the solution was selected as the best one during the cross-validation iterations. Right: Multivariate regression tree of the Doubs fish species explained by their environmental variables. Interpretation: see text

Let us build a multivariate regression tree of the Doubs fish data constrained by the environmental variables. We will use the fish species data with site vectors normalized to length 1 (chord transformation in Sect. 2.2.4) so that the distance actually preserved is the chord distance. Among the arguments, we will use `xval` = 29 cross-validation groups (i.e., as many groups as rows in the data, instead of the usual 10, owing to the small number of observations), 100 iterations, and we will allow ourselves to interactively pick a solution on a graph provided by **`mvpart()`** (Fig. 4.27, left). The graph displays the relative error (RE), which is steadily decreasing when the number of groups increases, and the cross-validated relative error (CVRE), which usually drops first sharply and then goes to a minimum before increasing again. Prediction is optimal for a given number of clusters, but its quality decreases when the data are exaggeratedly fragmented into many small groups.

The best solution is not always obvious, however. Sometimes it is the simple two-group solution that shows the smallest CVRE. The graph provides error bars representing one standard error for the CVRE, and an orange horizontal line located one standard error above the minimal CVRE solution (large red spot). According to De'ath (2002), one may select that tree as the best predictive tree or, following the rule proposed by Breiman et al. (1984) for univariate regression trees, one may select the smallest tree within one standard error of the best tree; this is the tree with k = 2 in our example, identified by "Min + 1 SE". That tree is more parsimonious and only slightly worse than the best predictive tree.

```
par(mfrow = c(1, 2))
spe.ch.mvpart <-
  mvpart(
    data.matrix(spe.norm) ~ .,
    env,
    margin = 0.08,
    cp = 0,
    xv = "pick",
    xval = nrow(spe),
    xvmult = 100
  )

summary(spe.ch.mvpart)
printcp(spe.ch.mvpart)
```

*Hint* *Argument* `xv = "pick"` *allows one to interactively pick a tree among those proposed. If one prefers that the tree with the minimum CVRE be automatically chosen, then use* `xv = "min"`.

If argument `xv = "pick"` has been used, which we recommend, one left-clicks on the point representing the desired number of groups. A tree is then drawn. Here we decided to pick the 4-group solution. While not the absolute best, it still ranks among the good ones and avoids producing too many small groups (Fig. 4.27, right).

The tree produced by this analysis is rich in information. Apart from the general statistics appearing at the bottom of the plot (residual error, i.e. the one-complement of the $R^2$ of the model; cross-validated error; standard error), the following features are important:

- Each node is characterized by a threshold value of an explanatory variable. For instance, the first node splits the data into two groups of 16 and 13 sites on the basis of elevation. The critical value (here 341 m) is often not found among the data; it is the mean of the two values delimiting the split. If two or more explanatory variables lead to equal results, an arbitrary choice is made among them. In this example, for instance, variable `dfs` with value 204.8 km would yield the same split.
- Each leaf (terminal group) is characterized by its number of sites and its RE as well as by a small barplot representing the abundances of the species (in the same order as is the response data matrix). Although difficult to read if there are many species, these plots show that the different groups are indeed characterized by different species. A more formal statistical approach is to search for characteristic or indicator species (Sect. 4.11). See below for an example.
- The tree can be used to allocate a new observation to one of the groups on the basis of the values of the relevant environmental variables. "Relevant" means here that the variables needed to allocate an object may differ depending on the

branch of the tree. Observe that a given variable may be used several times along the course of the successive binary partitions.

As proposed in the code below, apart from the residuals, one can retrieve the objects of each node and examine the node's characteristics at will:

```
# Residuals of MRT
par(mfrow = c(1, 2))
hist(residuals(spe.ch.mvpart), col = "bisque")
plot(predict(spe.ch.mvpart, type = "matrix"),
     residuals(spe.ch.mvpart),
     main = "Residuals vs Predicted")
abline(h = 0, lty = 3, col = "grey")

# Group composition
spe.ch.mvpart$where

# Group identity
(groups.mrt <- levels(as.factor(spe.ch.mvpart$where)))

# Fish composition of first leaf
spe.norm[which(spe.ch.mvpart$where == groups.mrt[1]), ]

# Environmental variables of first leaf
env[which(spe.ch.mvpart$where == groups.mrt[1]), ]
```

One can also use the MRT results to produce pie charts of the fish composition of the leaves (Fig. 4.28):

```
# Table and pie charts of fish composition of leaves
leaf.sum <- matrix(0, length(groups.mrt), ncol(spe))
colnames(leaf.sum) <- colnames(spe)
for (i in 1:length(groups.mrt))
{
  leaf.sum[i, ] <-
    apply(spe.norm[which(spe.ch.mvpart$where == groups.mrt[i]), ],
          2, sum)
}
leaf.sum

par(mfrow = c(2, 2))
for (i in 1:length(groups.mrt)){
  pie(which(leaf.sum[i, ] > 0),
      radius = 1,
      main = paste("leaf #", groups.mrt[i]))
}
```

Unfortunately, extracting other numerical results from an **`mvpart()`** object is no easy task. This is why our colleague Marie-Hélène Ouellette wrote a wrapper doing just that and providing a wealth of additional, useful information. The package

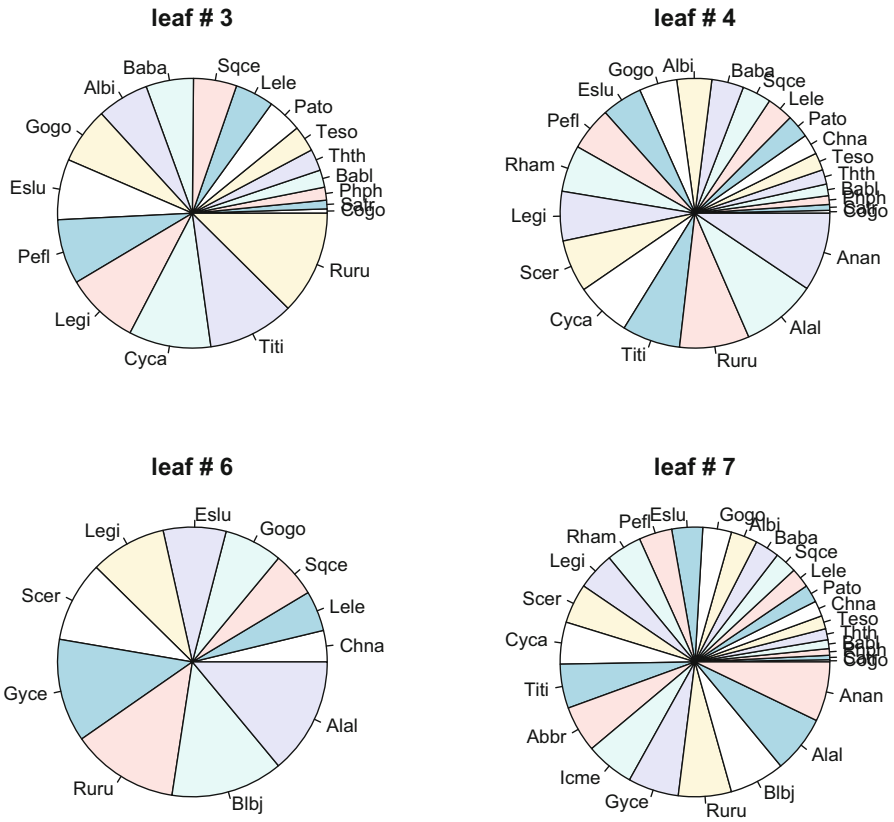


**Fig. 4.28** Pie chart of the fish composition of the four leaves retained in the MRT analysis

is called **`MVPARTwrap`** and the function is **`MRT()`**. Its output comprises two graphs and a lot of numerical results. Let us apply it to our previous result.

```
# Extracting MRT results from an mvpart object
spe.ch.mvpart.wrap <-
  MRT(spe.ch.mvpart, percent = 10, species = colnames(spe))
summary(spe.ch.mvpart.wrap)
```

*Hint* *Argument* `percent` *indicates the smallest percentage of variance explained by a species at a node that one considers interesting. No test is made here; it is an arbitrarily chosen value.*

The function displays some results on-screen during execution. These are presented in a form close to canonical ordination results (see Chap. 6), because the function actually runs a redundancy analysis on the species data, explained by the variables retained by the MRT analysis, and recoded following the threshold values of the tree nodes. Among the results, let us mention the $R^2$, which is the one-complement of the tree RE. In our example, Fig. 4.27 gives an error of 0.37 for the tree; therefore $R^2$ is equal to 0.63.

The summary of the result object provides information about the contribution of each node to the explained variance ("Complexity"). The sum of these values gives the overall $R^2$. Furthermore, it identifies "discriminant" species at each node, selecting the species that contribute most to the explained variance (down to a minimum arbitrarily set by the user with the argument `percent`). The mean (transformed) abundances are given for both branches of the nodes, so one can see the branch for which the species are discriminant. For instance, our example analysis shows species `Satr`, `Phph` and `Babl` for the left branch (higher altitude), and `Alal` for the right. The summary lists the sites present in each leaf. The result object itself contains the complete results from which the summary is drawn.

### 4.12.4 Combining MRT and IndVal

As suggested in the previous section, one could submit the MRT partition to a search for indicator species (IndVal, Sect. 4.11.2). This is better than visual examination of the results for the identification of "discriminant" species: one can test the significance of the indicator values and correct the p-values for multiple testing using the Holm correction.

```
# Indicator species search on the MRT result
spe.ch.MRT.indval <- indval(spe.norm, spe.ch.mvpart$where)
pval.adj3 <- p.adjust(spe.ch.MRT.indval$pval)     # Corrected prob.
```

```
 Cogo  Satr  Phph  Babl  Thth  Teso  Chna  Pato  Lele  Sqce  Baba  Albi
1.000 0.027 0.036 0.330 1.000 1.000 0.330 0.860 1.000 1.000 0.027 0.504
 Gogo  Eslu  Pefl  Rham  Legi  Scer  Cyca  Titi  Abbr  Icme  Gyce  Ruru
0.594 1.000 0.860 0.027 0.027 0.860 0.027 0.068 0.027 0.144 0.330 1.000
 Blbj  Alal  Anan
0.027 0.027 0.027
```

The following list gives the leaf number for the significant indicator species, followed by the list of the indicator values:

```
# For each significant species, find the leaf with the highest
# IndVal
spe.ch.MRT.indval$maxcls[which(pval.adj3 <= 0.05)]
```

```
Satr Phph Baba Rham Legi Cyca Abbr Blbj Alal Anan
   1    1    4    4    4    4    4    4    4    4
```

```
# IndVal value in the best leaf for each significant species
spe.ch.MRT.indval$indcls[which(pval.adj3 <= 0.05)]
```

```
      Satr       Phph       Baba      Rham       Legi       Cyca       Abbr
0.7899792 0.5422453 0.6547659 0.8743930 0.6568224 0.7964617 0.9000000
      Blbj       Alal       Anan
0.7079525 0.6700303 0.8460903
```

One sees that not all groups harbour indicator species, and that most of these are in the fourth (rightmost) group. Individually, the result for the brown trout (`Satr`), for instance, shows a significant indicator value of 0.79 in the first leaf.

Then, it would be interesting to compare the constrained typology of sites brought by MRT with the one obtained by the unconstrained optimized Ward clustering. For this purpose, we transform the leaf numbers to sequential numbers, i.e. the levels of the membership factor.

```
# Partition of objects based on MRT
spech.mvpart.g <- factor(spe.ch.mvpart$where)
levels(spech.mvpart.g) <- 1:length(levels(spech.mvpart.g))
# Compare with partition from unconstrained clustering
table(spech.mvpart.g, spech.ward.g)
```

Finally, the MRT clusters can be mapped on the Doubs River (Fig. 4.29):

```
# Plot of the MRT clusters on a map of the Doubs River
drawmap(xy = spa,
        clusters = spech.mvpart.g,
        main = "Four MRT clusters along the Doubs River")
```

## 4.13 MRT as a Monothetic Clustering Method

Multivariate regression trees provide a solution for monothetic clustering, which is a form of (unconstrained) clustering where a single response variable is selected for each split of the tree. The application here consists of building the tree using the species data as both the response and explanatory variables. This allows the definition of groups of sites based on a homogeneous composition and explained by the presence (or the relative abundance) of indicator species added sequentially. This method is related to the "association analysis" proposed by Williams and Lambert (1959). It can be applied to presence-absence or pre-transformed abundance response data; the explanatory data may be binary (presence-absence) or quantitative (untransformed or transformed), the same data matrix being used, possibly after

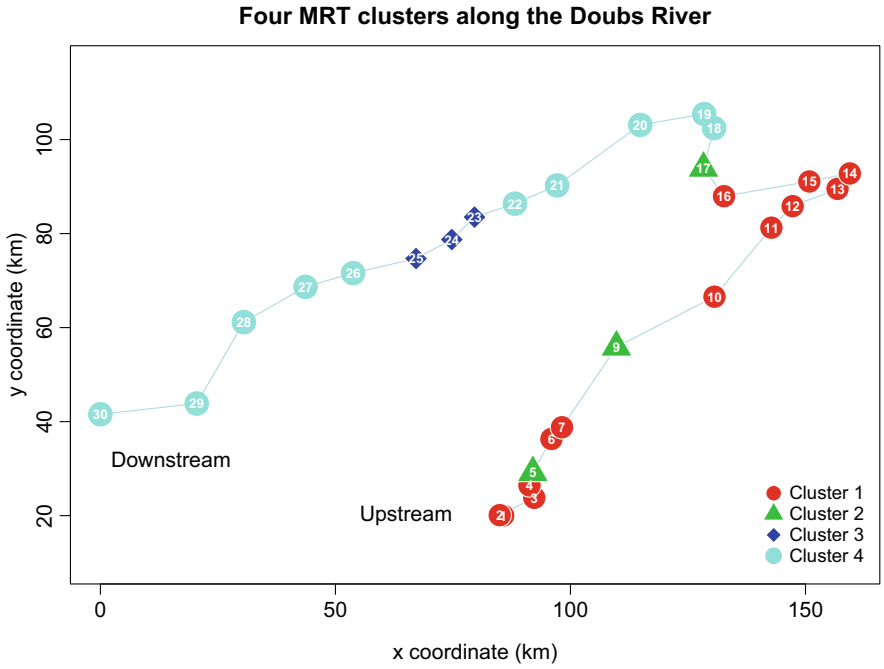


**Fig. 4.29** Four MRT clusters along the Doubs River

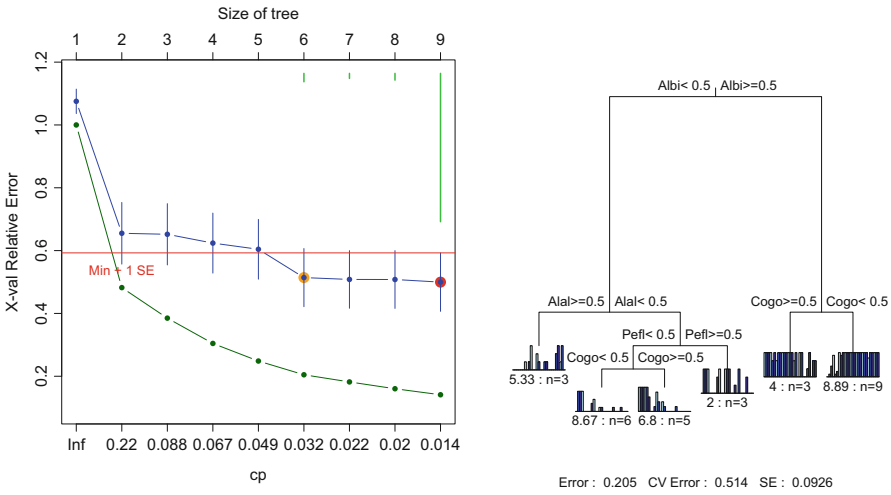


**Fig. 4.30** Monothetic clustering of the fish species data from MRT with the presence-absence dissimilarity matrix used both as response and explanatory matrix

different transformations, for both the explanatory and response data. Here we apply this method to the Doubs fish data (Figs. 4.30 and 4.31).

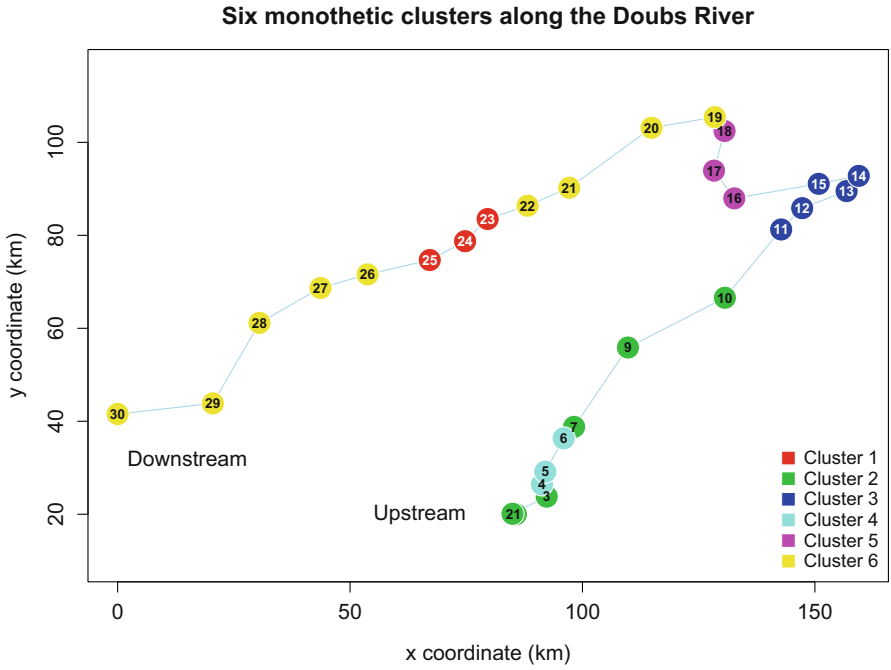


**Fig. 4.31** Six monothetic clusters along the Doubs River. The response and "explanatory" matrices are the presence-absence data

```
# spe.pa (presence-absence) is the response and the explanatory
# matrix
par(mfrow = c(1, 2))
spe.pa <- decostand(spe, "pa")
res.part1 <-
  mvpart(
    data.matrix(spe.pa) ~ .,
    data = spe.pa,
    margin = 0.08,
    xv = "p",
    xvmult = 100
  )

# spe.norm is the response, spe.pa is the explanatory matrix
res.part2 <-
  mvpart(
    data.matrix(spe.norm) ~ .,
    data = spe.pa,
    margin = 0.08,
    xv = "p",
    xvmult = 100
  )
```

```
# spe.norm is the response matrix and spe (untransformed) is the
# explanatory matrix
res.part3 <-
  mvpart(
    data.matrix(spe.norm) ~ .,
    data = spe,
    margin = 0.08,
    xv = "p",
    cp = 0,
    xvmult = 100
  )

# Membership of objects to groups - presence-absence on both sides
res.part1$where

res.part1.g <- factor(res.part1$where)
levels(res.part1.g) <- 1:length(levels(res.part1.g))
# Compare with groups from unconstrained clustering
table(res.part1.g, spech.ward.g)
table(res.part1.g, spech.ward.gk)

# Plot of the MRT clusters on a map of the Doubs River
drawmap3(xy = spa,
        clusters = res.part1.g,
        main = "Six monothetic clusters along the Doubs River")
```

## 4.14 Sequential Clustering

In cases where the data present themselves in a spatial (transect) or temporal sequence, the contiguity information can be taken into account when looking for groups, and for discontinuities, along the series. Several methods have been proposed for that purpose in the temporal (e.g. Gordon and Birks 1972, 1974; Gordon 1973, Legendre et al. 1985) and spatial (Legendre et al. 1990) contexts. Sequential clustering can also be computed by MRT, the constraint being a variable representing the sampling sequence (Legendre and Legendre 2012 Sect. 12.6.4). For the 29 sites of the Doubs data, a vector containing the numbers 1 to 29 or, equivalently, the variable `dfs`, would be appropriate. Computation of this example is detailed in Borcard et al. (2011, Sect. 4.11.5). The code is presented in the accompanying material of the present book.

Here we will apply a method of clustering with contiguity constraint developed for stratigraphic research and called CONISS (Grimm 1987). It can be construed as a variant of Ward's minimum variance clustering with the constraint that sites can only join if they are contiguous along a spatial or temporal sequence. The CONISS algorithm is available through function **`chclust()`** of package **`rioja`**.

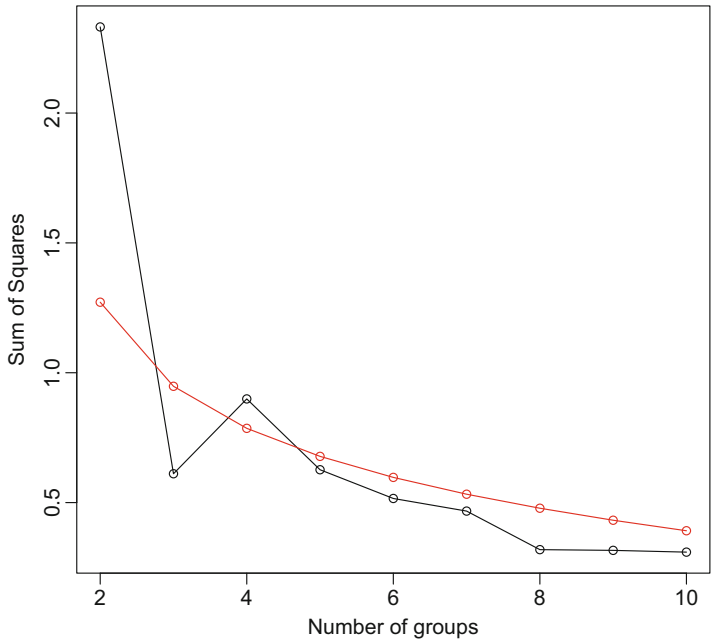


**Fig. 4.32** Group-by-SS graph comparing the dispersion of the classification at different fusion levels to a null model. Red line: broken stick model

We apply it to a matrix of percentage difference (aka Bray-Curtis) dissimilarities computed from the fish species data. To choose the number of clusters, the dispersion (sum of squares, SS) of the hierarchical classification is compared to that obtained from a broken stick model (Fig. 4.32):

```
# Default method CONISS
# On the percentage difference dissimilarity matrix
spe.chcl <- chclust(vegdist(spe))

# Compare the fusion levels to a broken stick model
bstick(spe.chcl, 10)
```

*The graph suggests cutting the dendrogram to retain two or four clusters (points above the red line obtained from the broken stick model).*

For consistency with previous clustering results, we choose to cut the dendrogram in 4 groups. The result is displayed in Fig. 4.33 as a dendrogram and in the form of a map of the four clusters along the river.

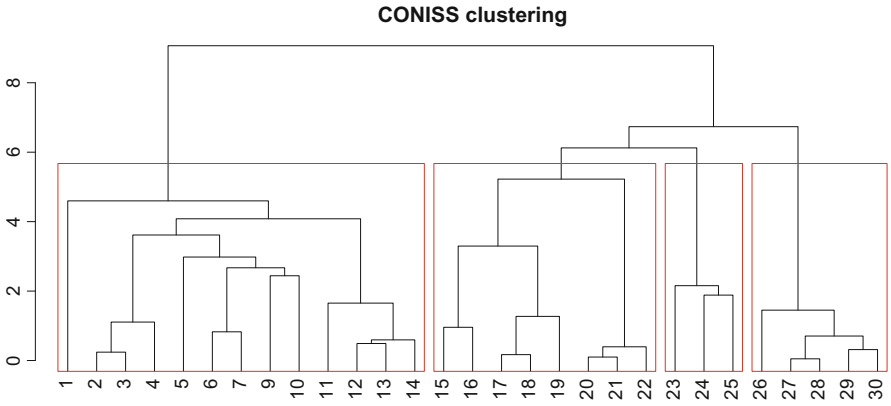


**Fig. 4.33** Dendrogram of the clustering with contiguity constraint of the fish species data

```
# Cut the dendrogram in 4 clusters
k <- 4
(gr4 <- cutree(spe.chcl, k = k))

# Plot the dendrogram
plot(spe.chcl, hang = -1, main = "CONISS clustering")
rect.hclust(spe.chcl, k = k)
```

*If you want, you can space out the branches of the dendrogram according to the distance from the source:*

```
# Dendrogram with observations plotted according to dfs
plot(spe.chcl,
     xvar = env$dfs,
     hang = -1,
     main = "CONISS clustering",
     cex = 0.8)
```

See on the map of the Doubs River how sites are now clustered in a consistent sequence. The constraint of spatial contiguity has forced the separation of sites 15–22 and 26–30 into separate groups because the sequence is interrupted by the group 23–25 of polluted sites (Fig. 4.34):

```
# Plot the clusters on a map of the Doubs River
drawmap(xy = spa,
        clusters = gr4,
        main = "Sequential clusters along the river")
```

*Compare this sequence with the traditional zonation of fish communities presented in Chap.1.*

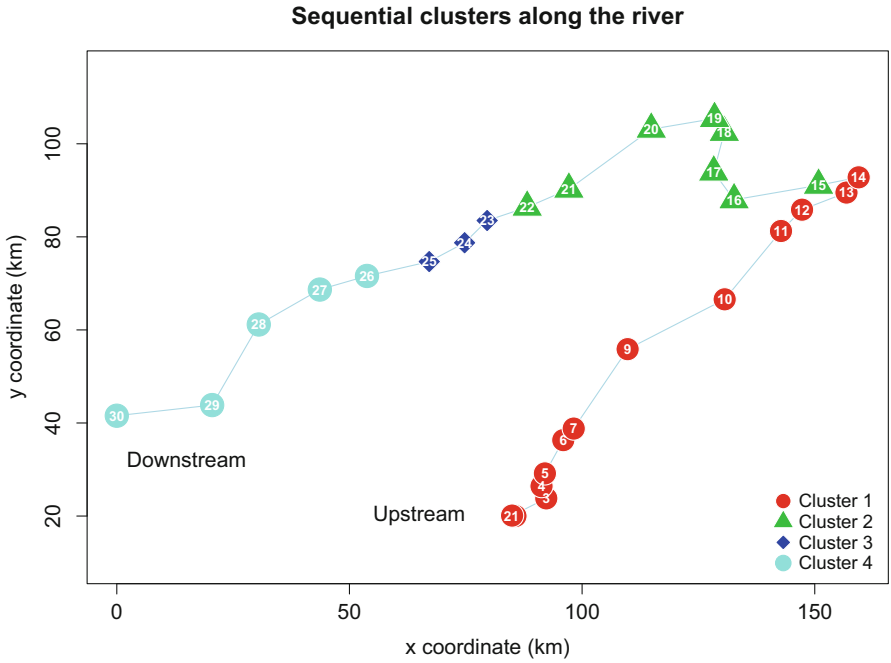


**Fig. 4.34** Map of four groups along the Doubs River

## 4.15 A Very Different Approach: Fuzzy Clustering

At the beginning of this chapter, we defined clusters produced by clustering methods as non-overlapping entities. This definition is a natural consequence of the focus of most clustering methods on discontinuities. However, there is another approach to clustering that recognizes that sometimes cluster limits may not be so clear-cut as one would like them to be. In that approach, an object may be, to different degrees, a member of two or several groups. As an example, the colour green may be obtained by mixing blue and yellow paint, the proportion of each of the two primary colours determining the shade of green. Consequently, a different family of hierarchical and non-hierarchical methods has been developed, namely fuzzy clustering. We will not develop this family in detail, but briefly show one approach that is akin to non-hierarchical *k*-means partitioning. Its name is *c*-means clustering (Kaufman and Rousseeuw 2005).

### *4.15.1 Fuzzy c-means Using Package* `cluster`*'s Function* `fanny()`

Instead of a classification where a given object belongs to one and only one cluster, *c*-means clustering associates to all objects a series of membership values measuring

the strength of their memberships in the various clusters. An object that is clearly linked to a given cluster has a strong membership value for that cluster and weak (or null) values for the other clusters. The membership values add up to 1 for each object. To give an example from sensory evaluation, imagine the difficulty of applying the two-state descriptor {sweet, bitter} to beers. A beer may be judged by a panel of tasters to be, say, 30% sweet and 70% bitter. This is an example of a fuzzy classification of beers for that descriptor.

Fuzzy *c*-means clustering is implemented in several packages, e.g. **`cluster`** (function **`fanny()`**) and **`e1071`** (function **`cmeans()`**). The short example below uses the former.

Function **`fanny()`** accepts either site-by-species or dissimilarity matrices. In the former case, the default metric is `euclidean`. Here we will directly use as input the chord distance matrix '`spe.ch`' previously computed from the fish species data. An identical result would be obtained by using the chord-transformed species data '`spe.norm`' with the `metric = "euclidean"` argument.

The plot function can return two diagrams: an ordination (see Chap. 5) of the clusters and a silhouette plot. Here we present the latter (Fig. 4.35) and we replace the original ordination diagram by a principal coordinate analysis (PCoA, see Sect. 5.5) combined with star plots of the objects (Fig. 4.36). Each object is associated with a small star plot (resembling a little pie chart) whose segment radiuses are proportional to its membership coefficient.

```
k <- 4      # Choose the number of clusters
spe.fuz <- fanny(spe.ch, k = k, memb.exp = 1.5)
summary(spe.fuz)

# Site fuzzy membership
spe.fuz$membership
# Nearest crisp clustering
spe.fuz$clustering
spefuz.g <- spe.fuz$clustering

# Silhouette plot
plot(
  silhouette(spe.fuz),
  main = "Silhouette plot - Fuzzy clustering",
  cex.names = 0.8,
  col = spe.fuz$silinfo$widths + 1
)

# Ordination of fuzzy clusters (PCoA)
# Step 1: ordination (PCoA) of the fish chord distance matrix
dc.pcoa <- cmdscale(spe.ch)
dc.scores <- scores(dc.pcoa, choices = c(1, 2))
# Step 2: ordination plot of fuzzy clustering result
plot(dc.scores,
     asp = 1,
     type = "n",
     main = "Ordination of fuzzy clusters (PCoA)")
```

```
abline(h = 0, lty = "dotted")
abline(v = 0, lty = "dotted")
# Step 3: representation of fuzzy clusters
for (i in 1:k){
  gg <- dc.scores[spefuz.g == i, ]
  hpts <- chull(gg)
  hpts <- c(hpts, hpts[1])
  lines(gg[hpts, ], col = i + 1)
}
stars(
  spe.fuz$membership,
  location = dc.scores,
  key.loc = c(0.6, 0.4),
  key.labels = 1:k,
  draw.segments = TRUE,
  add = TRUE,
  # scale = FALSE,
  len = 0.075,
  col.segments = 2:(k + 1)
)
```

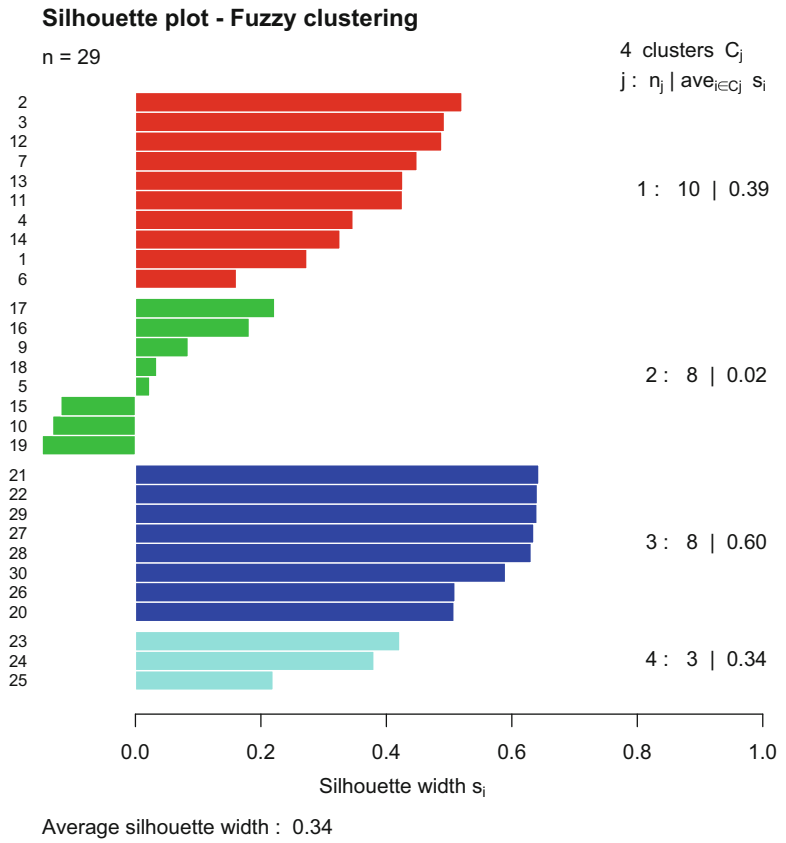


**Fig. 4.35** Silhouette plot of the *c*-means fuzzy clustering of the fish data preserving the chord distance

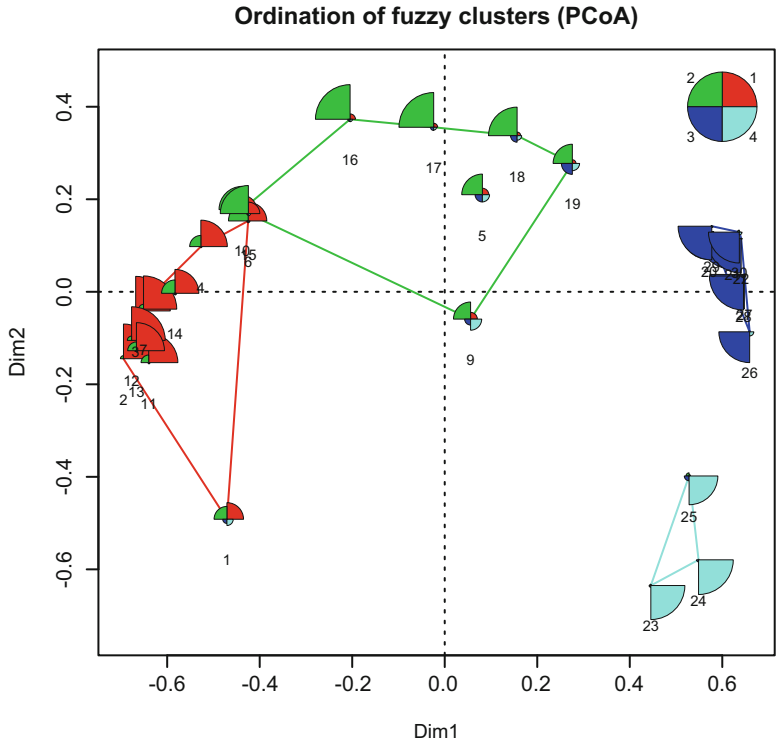


**Fig. 4.36** *c*-means fuzzy clustering of the fish data preserving the chord distance. Principal coordinate ordination associated with star plots showing the memberships of the sites

*Hints* *Argument* `memb.exp` *is a kind of "fuzziness exponent" with values ranging from 1 (close to non-fuzzy clustering) to any large value.*

The silhouette plot (Fig. 4.35) shows, in particular, that cluster 2 is not well defined. On the ordination diagram (Fig. 4.36), the star plots of the ill-classified objects (10, 15, 19) also illustrate that their membership is unclear.

The numerical results give the membership coefficients of the objects. The sum of each row is equal to 1. Objects belonging unambiguously to one cluster, like sites 2, 21 or 23, have a high membership value for that cluster and correspondingly low values for the other clusters. Conversely, one can easily locate objects that are difficult to classify: their coefficients have similar values in most if not all clusters. Sites 5, 9 and 19 are good examples. An additional result is the nearest crisp clustering, i.e., the cluster to which each object has the highest membership coefficient.

Sectors representing fuzzy membership can also be added to the map of the sites along the Doubs River (Fig. 4.37):

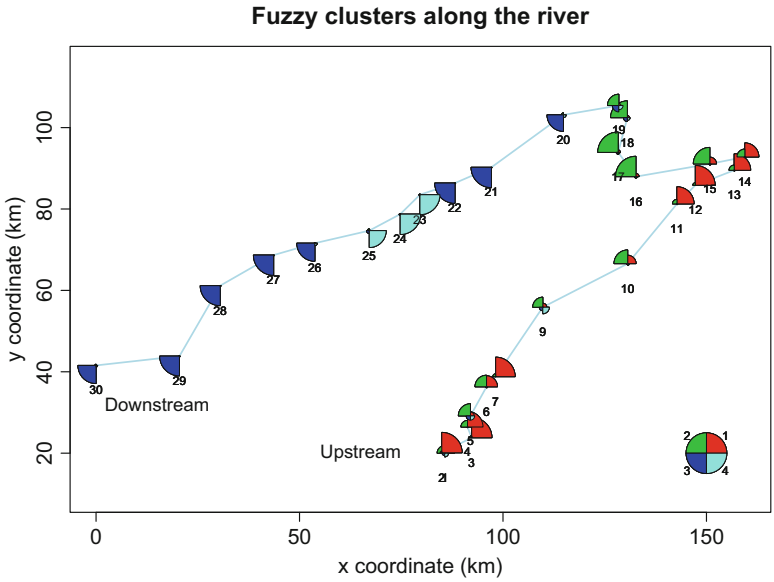


**Fig. 4.37** Four fuzzy clusters along the Doubs River

```
# Plot the fuzzy clusters on a map of the Doubs River
plot(
  spa,
  asp = 1,
  type = "n",
  main = "Fuzzy clusters along the river",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
text(65, 20, "Upstream", cex = 1.2)
text(15, 32, "Downstream", cex = 1.2)
# Add sectors to represent fuzzy membership
for (i in 1:k) {
  stars(
    spe.fuz$membership,
    location = spa,
    key.loc = c(150, 20),
    key.labels = 1:k,
    draw.segments = TRUE,
    add = TRUE,
    len = 5,
    col.segments = 2:(k + 1)
  )
}
```

### 4.15.2 Noise Clustering Using the `vegclust()` Function

A recent package called **`vegclust`**, developed by Miquel De Cáceres, provides a large range of options to perform non-hierarchical or hierarchical fuzzy clustering of community data under different models (De Cáceres et al. 2010). An interesting one is called "noise clustering" (Davé and Krishnapuram 1997). It allows the consideration of outliers, i.e. unclassified objects, denoted "N", beside fuzzy clusters labelled "M1", "M2"... The principle of the method consists in defining a cluster called "Noise" in addition to the regular clusters. This "Noise" cluster is represented by an imaginary point located at a constant distance δ from all observations. The effect of this cluster is to capture the "*objects that lie farther than* δ *from all the c "good" centroids*" (De Cáceres et al. 2010). A small δ results in a large membership in the "Noise" cluster.

To perform noise clustering, we apply the **`vegclust()`** function to the normalized species matrix with the argument `method = "NC"`. As before, we project the result of the noise clustering on a PCoA plot using sectors to represent fuzzy membership (Fig. 4.38).

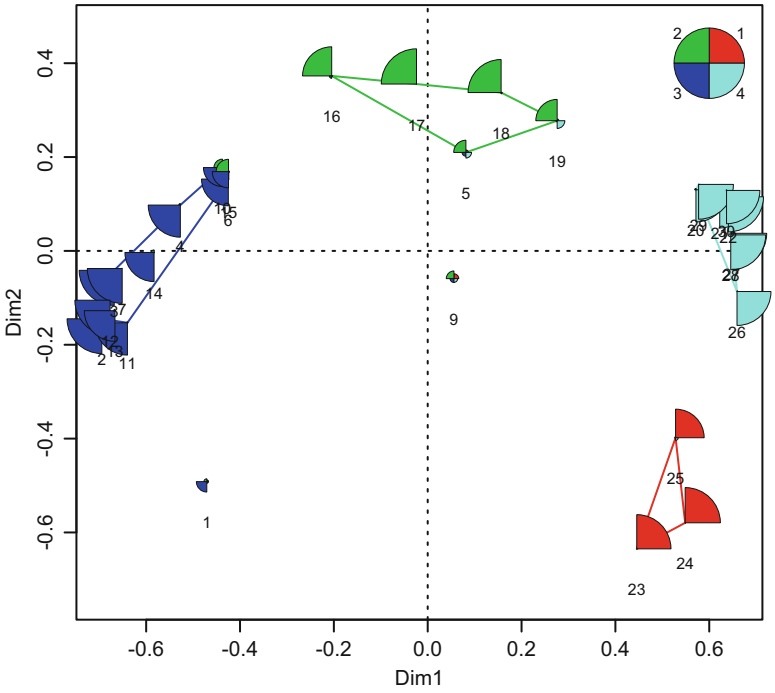


**Fig. 4.38** Ordination plot of the noise clustering of the fish species data.

```
# Create noise clustering with four clusters. Perform 30 starts
# from random seeds and keep the best solution
k <- 4
spe.nc <- vegclust(
  spe.norm,
  mobileCenters = k,
  m = 1.5,
  dnoise = 0.75,
  method = "NC",
  nstart = 30
)
spe.nc

# Medoids of species
(medoids <- spe.nc$mobileCenters)

# Fuzzy membership matrix
spe.nc$memb

# Cardinality of fuzzy clusters (i.e., the number of objects
# belonging to each cluster)
spe.nc$size

# Obtain hard membership vector, with 'N' for objects that are
# unclassified
spefuz.g <- defuzzify(spe.nc$memb)$cluster
clNum <- as.numeric(as.factor(spefuz.g))

# Ordination of fuzzy clusters (PCoA)
plot(dc.scores,
     asp = 1,
     type = "n")
abline(h = 0, lty = "dotted")
abline(v = 0, lty = "dotted")
for (i in 1:k)
{
  gg <- dc.scores[clNum == i, ]
  hpts <- chull(gg)
  hpts <- c(hpts, hpts[1])
  lines(gg[hpts, ], col = i + 1)
}
stars(
  spe.nc$memb[, 1:4],
  location = dc.scores,
  key.loc = c(0.6, 0.4),
  key.labels = 1:k,
  draw.segments = TRUE,
  add = TRUE,
  len = 0.075,
  col.segments = 2:(k + 1)
  )
```

The hard partition contained in the **`clNum`** object is added to the ordination plot (Fig. 4.39).

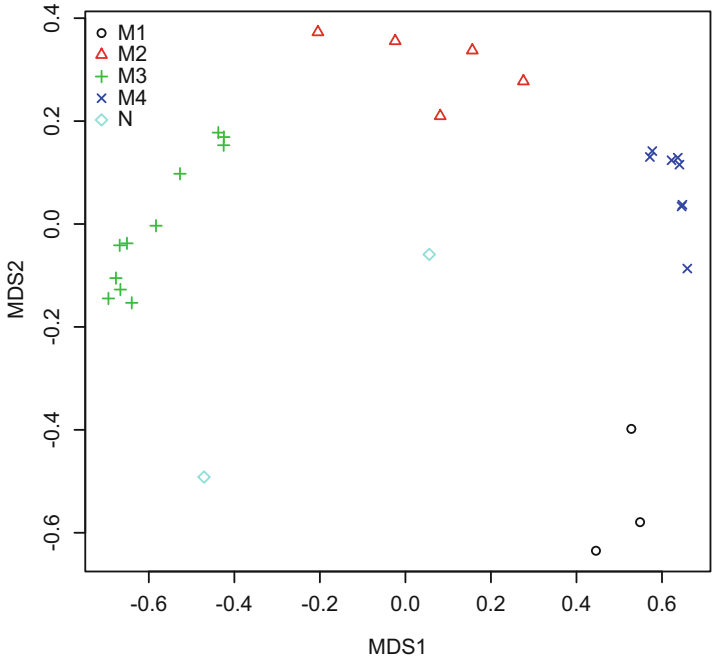


**Fig. 4.39** Ordination plot of the defuzzified noise clustering of the fish species data, showing the two unclassified objects

```
# Defuzzified site plot
plot(
  dc.pcoa,
  xlab = "MDS1",
  ylab = "MDS2",
  pch = clNum,
  col = clNum
)
legend(
  "topleft",
  col = 1:(k + 1),
  pch = 1:(k + 1),
  legend = levels(as.factor(spefuz.g)),
  bty = "n"
)
```

Finally, let us plot the fuzzy clusters on the map of the river (Fig. 4.40).

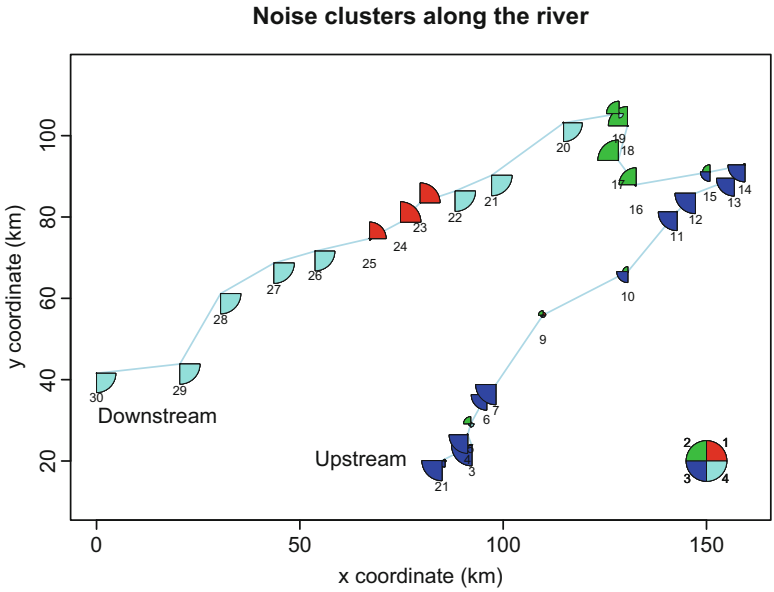


**Fig. 4.40** Four noise clusters along the Doubs River

```
# Plot the fuzzy clusters on a map of the Doubs River
plot(
  spa,
  asp = 1,
  type = "n",
  main = "Noise clusters along the river",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
text(65, 20, "Upstream", cex = 1.2)
text(15, 32, "Downstream", cex = 1.2)
# Add sectors to represent fuzzy membership
for (i in 1:k) {
  stars(
    spe.nc$memb[, 1:4],
    location = spa,
    key.loc = c(150, 20),
    key.labels = 1:k,
    draw.segments = TRUE,
    add = TRUE,
    # scale = FALSE,
    len = 5,
    col.segments = 2:(k + 1)
  )
}
```

While “hard” clustering may appear somewhat unrealistic in ecology, its application is of great help when one needs a typology or a decision-making tool requiring unambiguous allocation of the sites. Fuzzy clustering is a more nuanced, and therefore more realistic approach in most ecological situations if its purpose is to describe similarity relationships among sites. Other very powerful methods exist that are designed to reveal the structure of continuous data. These methods, simple and constrained ordination, are explored in the following chapters.

## 4.16 Conclusion

This chapter has not covered all the possibilities offered by the large family of cluster analyses, but you have explored its main avenues and seen how flexible this approach can be. Every ecological research project has its own features and constraints; in many cases cluster analysis can provide rich insights into the data. The clustering techniques themselves are numerous, as are also the ways of interpreting the results. It is up to you to exploit this lore to optimise the output of your research.