Use R!

Series Editors:
Robert Gentleman Kurt Hornik Giovanni G. Parmigiani

Use R!

Albert: Bayesian Computation with R
Bivand/Pebesma/Gómez-Rubio: Applied Spatial Data Analysis with R
Cook/Swayne: Interactive and Dynamic Graphics for Data Analysis:
With R and GGobi
Hahne/Huber/Gentleman/Falcon: Bioconductor Case Studies
Paradis: Analysis of Phylogenetics and Evolution with R
Pfaff: Analysis of Integrated and Cointegrated Time Series with R
Sarkar: Lattice: Multivariate Data Visualization with R
Spector: Data Manipulation with R

Daniel Borcard · François Gillet
Pierre Legendre

Numerical Ecology with R

Springer

Daniel Borcard
Département de sciences biologiques
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal, Québec
H3C 3J7 Canada
daniel.borcard@umontreal.ca

Pierre Legendre
Département de sciences biologiques
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal, Québec
H3C 3J7 Canada
pierre.legendre@umontreal.ca

François Gillet
Université de Franche-Comté - CNRS
UMR 6249 Chrono-environnement
UFR Sciences et Techniques
16, Route de Gray
F-25030 Besançon cedex
France
francois.gillet@univ-fcomte.fr

Series Editors:
Robert Gentleman
Program in Computational Biology
Division of Public Health Sciences
Fred Hutchinson Cancer Research Center
1100 Fairview Ave. N, M2-B876
Seattle, Washington 98109-1024
USA

Kurt Hornik
Department für Statistik und Mathematik
Wirtschaftsuniversität Wien Augasse 2-6
A-1090 Wien
Austria

Giovanni G. Parmigiani
The Sidney Kimmel Comprehensive Cancer
Center at Johns Hopkins University
550 North Broadway
Baltimore, MD 2105-2011
USA

ISBN 978-1-4419-7975-9 e-ISBN 978-1-4419-7976-6
DOI 10.1007/978-1-4419-7976-6
Springer New York Dordrecht London Heidelberg

Preface

Ecology is sexy. Teaching ecology is therefore the art of presenting a fascinating topic to well-predisposed audiences. It is not easy: the complexities of modern ecological science go well beyond the introductory chapters taught in high schools or the marvellous movies about ecosystems presented on TV. But well-predisposed audiences are ready to make the effort. *Numerical* ecology is another story. For some unclear reasons, a majority of ecology-oriented people are strangely reluctant when it comes to quantifying nature and using mathematical tools to help understand it. As if nature was inherently non-mathematical, which it is certainly not: mathematics is the common language of all sciences. Teachers of biostatistics and numerical ecology thus have to overcome this reluctance: before even beginning to teach the subject itself, they must convince their audience of the interest and necessity of it.

During many decades, ecologists, be they students or researchers (in the academic, private or government spheres), used to plan their research and collect data with few, if any, statistical consideration, and then entrusted the "statistical" analyses of their results to a person hired especially for that purpose. That person may well have been a competent statistician, and indeed in many cases, the progressive integration of statistics into the whole process of ecological research was triggered by such people. In other cases, however, the end product was a large amount of data summarized using a handful of basic statistics and tests of significance that were far from revealing all the richness of the structures hidden in the data tables. The separation of the ecological and statistical worlds presented many problems. The most important were that the ecologists were unaware of the array of methods available at the time, and the statisticians were unaware of the ecological hypotheses to be tested and the specific requirements of ecological data (the double-zero problem is a good example). Apart from preventing the data to be exploited properly, this double unawareness prevented the development of methods specifically tailored to ecological problems.

The answer to this situation is to form mathematically inclined ecologists. Fortunately, more and more such people have appeared during the past four decades. The result of their work is a huge development of statistical ecology, the availability of several excellent textbooks and the increasing awareness of the

responsibility of ecologists with regard to the proper design and analysis of their research. This awareness makes the task easier for teachers as well.

Until relatively recently, however, a critical ingredient was still missing for the teaching to be efficient and for the practice of statistics to become generalized among ecologists: a set of standard packages available to everyone, everywhere. A biostatistics or numerical ecology course means nothing without practical exercises. A course linked to a commercial software is much better, but it is bound to restrict future applications if the researcher moves and loses access to the software that he or she knows. Furthermore, commercial packages are in most cases written for larger audiences than the community of ecologists and they may not include all the functions required for analysing ecological data. The **R** language resolved that issue, thanks to the dedication of the many researchers who created and freely contributed extensive, well-designed and well-documented packages. Now, the teacher no longer has to say: "this is the way PCA works… on paper"; she or he can say instead: "this is the way PCA works, now I show you on-screen how to run one, and in a few minutes you will be able to run your own, and do it anywhere in the world on your own data!".

Another fundamental property of the **R** language is that it is meant as a self-learning environment. A book on **R** is therefore bound to follow that philosophy, and must provide the support necessary for anyone wishing to explore the subject by himself or herself. This book has been written to provide a bridge between the theory and practice of numerical ecology that anyone can cross. Our dearest hope is that it will make many happy teachers and happy ecologists.

Montréal, QC — Daniel Borcard
Besançon, France — François Gillet
Montréal, QC — Pierre Legendre

Contents

Chapter 1
Introduction

1.1 Why Numerical Ecology?

Although multivariate analysis of ecological data already existed and was being actively developed in the 1960s, it really flourished in the years 1970 and later. Many textbooks were published during these years; among them were the seminal *Écologie numérique* (Legendre and Legendre 1979) and its English translation *Numerical Ecology* (Legendre and Legendre 1983). The authors of these books unified, under one single roof, a very wide array of statistical and other numerical techniques and presented them in a comprehensive way, not only to help researchers understand the available methods of analyses, but also to explain how to choose and apply them in an ordered, logical way to reach their research goals. Mathematical explanations are not absent from these books, and they provide a precious insider look into the various techniques, which is appealing to readers wishing to go beyond the simple user level.

Since then, numerical ecology has become ubiquitous. Every serious researcher or practitioner has become aware of the tremendous interest of exploiting the painfully acquired data as efficiently as possible. Other manuals have been published (e.g. Orlóci and Kenkel 1985; Jongman et al. 1995; McCune and Grace 2002; McGarigal et al. 2000; Zuur et al. 2007). A second English edition of *Numerical Ecology* was published in 1998, broadening the perspective and introducing numerous methods that were unavailable at the time of the previous editions. The progress continues, and since 1998, many important breakthroughs have occurred. In the present book, we present some of these developments that we consider most important, albeit in a more user-oriented way than in the above mentioned manuals, using the **R** language. For the most recent methods, we provide explanations at a more fundamental level when we consider it appropriate and helpful.

Not all existing methods of data analysis are addressed in the book, of course. Apart from the most widely used and fruitful methods, our choices are based on our own experience as quantitative community ecologists. However, small sections have sometimes been added to briefly describe other avenues than the main ones, without going into details.

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_1, © Springer Science+Business Media, LLC 2011

1.2 Why R?

The **R** language has experienced such a tremendous development and reached such a wide array of users during the recent years that a justification of its application to numerical ecology is not required. Development also means that more and more domains of numerical ecology are now covered, up to the point where, computationally speaking, some of the most recent methods are actually only available through **R** packages.

This book is not intended as a primer in **R**, however. To find that kind of support, readers should consult the CRAN Web page (http://www.R-project.org). The link to *Manuals* provides many free electronic documents and the link to *Books* many references. Readers are expected to have a minimal working knowledge of the basics of the language, e.g. formatting data and importing them into **R**, awareness of the main classes of objects handled in this environment (vectors, matrices, data frames and factors), as well as the basic syntax necessary to manipulate, create, and otherwise use objects within **R**. Nevertheless, Chap. 2 starts at an elementary level as far as multivariate objects are concerned, since these are the main targets of most analyses addressed throughout the book, while not necessarily being most familiar to many users.

The book is by far not exhaustive as to the array of functions devoted to any of the methods. Usually, we present one or several variants, but often other functions are available in **R**. Centring the book on a small number of well-integrated packages and adding some functions of our own, when necessary, helps users up the learning curve while keeping the amount of package-level idiosyncrasies at a reasonable level. Our choices do not imply that other existing packages are inferior to the ones used in the book.

1.3 Readership and Structure of the Book

The intended audience of this book is the researchers, practitioners, graduate students and teachers who already have a background in general and multivariate statistics and wish to apply their knowledge to their data using the **R** language, as well as people willing to accompany their learning of the discipline with practical applications. Although an important part of this book follows the organization and symbolism of Legendre and Legendre (1998) and many references to that book are made herein, readers may draw their training from other sources without problem.

Combining an application-oriented book such as this one with a detailed exposé of the methods used in numerical ecology would have led to an impossibly long and cumbersome opus. However, all chapters start with a short introduction summarizing its subject matter, to ensure that readers are aware of the scope of the chapter and can appreciate the point of view from which the methods are addressed. Depending on the amount of documentation already existing in statistical textbooks, some introductions are longer than others.

Overall, the book guides readers through an applied exploration of the major methods of multivariate data analysis, as seen through the eye of an ecologist. Starting with some exploratory approaches (Chap. 2), it proceeds logically with the construction of the key building blocks of most techniques, i.e. association measures and matrices (Chap. 3), and then submits example data to three families of approaches: clustering (Chap. 4), ordination and canonical ordination (Chaps. 5 and 6), and finally spatial analysis (Chap. 7). The aims of methods thus range from descriptive to explanatory and to predictive and encompass a wide variety of approaches that should provide readers with an extensive toolbox that can address a wide palette of questions arising in contemporary multivariate ecological analysis.

1.4 How to Use This Book

The book is meant as a companion when working at the computer. The authors pictured a reader studying a chapter by reading the text and simultaneously executing the code. To fully understand the various methods, it is preferable to go through the chapters sequentially, since each builds upon the previous ones. At the beginning of each chapter, an empty **R** console is assumed to be open. All the necessary data files, the scripts used in the chapters, as well as the **R** functions and packages that are not available through the CRAN Web site, can be downloaded from a Web page accessible through the Springer Web site (http://www.springer.com/978-1-4419-7975-9). Some of the homemade functions duplicate existing ones, providing alternative solutions (for instance, different or expanded graphical outputs), while others have been written to streamline complex sequences of operations.

Although the code provided can be run in one single copy-and-paste shot within each chapter (with some rare exceptions for interactive functions), the best procedure is to proceed through the code slowly and explore each set of commands carefully. Although the use and meaning of some arguments is explained within the code or in the text, readers are warmly invited to use and abuse of the **R** help files (function name following a question mark) to learn about and explore the various options available. Our aim is not to describe all options of all functions, which would be an impossible and useless task. We are confident that an avid user, willing to go beyond the provided examples, will be kept busy for months exploring the options that he or she deems the most interesting.

Within each chapter, after the introduction, readers are invited to import the data for the exercises, as well as the **R** packages necessary for the whole chapter. The **R** code used in each chapter is self-contained, i.e. it can be run in one step without resorting to intermediate results produced in previous chapters. If such objects are needed, they are recomputed at the beginning of the chapter.

In everyday use, one generally does not produce an **R** object for every single operation, nor does one create and name a new graphical window for every plot. We do that in the **R** scripts to provide readers with all the entities necessary to backtrack the procedures, compare results and explore variants. Therefore, after having run

most of the code in a chapter, if one decides to explore another path using some intermediate result, the corresponding object will be available without the need to recompute it. This is particularly handy for results of computer-intensive methods (like some based on large numbers of random permutations), especially if one uses a relatively slow computer.

In the code sections of the book, all calls to graphical windows have been deleted for brevity. They are found in the electronic code scripts, however. Furthermore, the book shows several, but not all, graphical outputs for reference. They are printed in grey scale, although some are in colour when produced by **R**. This is an incentive for readers to be active users of the book and of its code.

Sometimes, readers are made aware of some special features of the code or of tricks used to obtain particular results, by means of hint boxes located at the bottom of code sections.

Although many methods are applied to the example data, ecological interpretations is not provided in all cases. Sometimes, questions are left open to readers, as an incentive to verify if she or he has correctly understood the method, and hence its application and the numerical or graphical outputs.

Lastly, for some methods programming-oriented readers are invited to write their own code. These incentives are placed in boxes called "code-it-yourself corners". When examples are readily written, they are meant for pedagogical purposes and do not pretend at computational efficiency. The aim of these boxes is to help interested readers code the matrix algebra equations presented in Legendre and Legendre (1998) into **R** and obtain the main outputs that ready-made packages provide. The whole idea is, of course, to reach the deepest possible understanding of the mathematical working of some key methods.

1.5 The Data Sets

Apart from rare cases where ad hoc fictitious data are built for special purposes, the applications rely on two main data sets that are readily available in **R**. However, data provided in **R** packages can change over the years. Therefore, we prefer to provide them also in the electronic material accompanying this book because this ensures that the results obtained by the readers are exactly the same as those presented in the book. The two data sets are briefly presented here. The first (Doubs) data set is explored in more detail in Chap. 2, and readers are encouraged to apply the same exploratory methods to the second one.

1.5.1 The Doubs Fish Data

In an important doctoral thesis, Verneaux (1973; see also Verneaux et al. 2003) proposed to use fish species to characterize ecological zones along European rivers and streams. He showed that fish communities were good biological indicators of

Table 1.1 Environmental variables of the Doubs data set used in this book and their units

Variable	Code	Units
Distance from source	`das`	km
Altitude	`alt`	m a.s.l.
Slope	`pen`	‰
Mean minimum discharge	`deb`	$m^3 s^{-1}$
pH of water	`pH`	–
Calcium concentration (hardness)	`dur`	$mg L^{-1}$
Phosphate concentration	`pho`	$mg L^{-1}$
Nitrate concentration	`nit`	$mg L^{-1}$
Ammonium concentration	`amm`	$mg L^{-1}$
Dissolved oxygen	`oxy`	$mg L^{-1}$
Biological oxygen demand	`dbo`	$mg L^{-1}$

these water bodies. Starting from the source, Verneaux proposed a typology in four zones, and he named each one after a characteristic species: the trout zone (from the brown trout *Salmo trutta fario*), the grayling zone (from *Thymallus thymallus*), the barbel zone (from *Barbus barbus*) and the bream zone (from the common bream *Abramis brama*). The two upper zones are considered as the "Salmonid region" and the two lowermost ones constitute the "Cyprinid region". The corresponding ecological conditions, with much variation among rivers, range from relatively pristine, well oxygenated and oligotrophic to eutrophic and oxygen-deprived waters.

The Doubs data set that is used in the present book consists of three matrices containing part of the data used by Verneaux for his studies. These data have been collected at 30 sites along the Doubs River, which runs near the France–Switzerland border in the Jura Mountains. The first matrix contains coded abundances of 27 fish species, the second matrix contains 11 environmental variables related to the hydrology, geomorphology and chemistry of the river, and the third matrix contains the geographical coordinates (Cartesian, X and *Y*) of the sites. These data have already served as test cases in the development of numerical techniques (Chessel et al. 1994).

Working with the environmental data available in the **R** package **`ade4`** (at the time of this writing, i.e. **`ade4`** version 1.4-14), we corrected a mistake in the `das` variable and restored the variables to their original units, which are presented in Table 1.1.

1.5.2 The Oribatid Mite Data

Oribatid mites (Acari: Oribatida) are a very diversified group of small (0.2–1.2 mm) soil-dwelling, mostly microphytophagous and detritivorous arthropods. A well-aerated soil or a complex substrate like *Sphagnum* mosses present in bogs and wet forests can harbour up to several hundred thousand (10^5) individuals per square

metre. Local assemblages are sometimes composed of over a hundred species, including many rare ones. This diversity makes oribatid mites an interesting target group to study community–environment relationships at very local scales.

The example data set is composed of 70 cores of mostly *Sphagnum* mosses collected on the territory of the *Station de biologie des Laurentides* of Université de Montréal, Québec, Canada in June 1989. The data were collected in order to test various ecological hypotheses about the relationships between living communities

Table 1.2 Environmental variables of the oribatid mite data set used in this book and their units

Variable	Code	Units
Substrate density (dry matter)	`SubsDens`	g dm^{-3}
Water content	`WatrCont`	g dm^{-3}
Substrate	`Substrate`	7 unordered classes
Shrubs	`Shrub`	3 ordered classes
Microtopography	`Topo`	Blanket – Hummock

Table 1.3 Several help resources in **R**

Action	Use	Example	Remark
`? (question mark)`	Obtain information about a function	`?decostand`	The package to which the function belongs must be active
`?? (double question mark)`	Obtain information on the basis of a keyword	`??diversity`	The search is done in all the packages installed in the computer
Type the name of a function	Display the code of the function on-screen	`diversity`	Not all functions can be displayed fully; some contain compiled code
`help (package="....")`	Displays information on the package, including a list of all functions and data	`help (package="ade4")`	
`data (package="...")`	Lists the data sets contained in a package	`data (package="vegan")`	
Search on the CRAN Web site	Broader search than above; access to discussion lists	Go to http://cran.r-project.org/, click on the "Search" link and choose one of the links	Outside the **R** master Web server

and their environment when the latter is spatially structured, and develop statistical techniques for the analysis of the spatial structure of living communities. It has since become a classical test data set, used in several publications (e.g. Borcard et al. 1992, 2004; Borcard and Legendre 1994; Wagner 2004; Legendre 2005; Dray et al. 2006; Griffith and Peres-Neto 2006).

The data set comprises three files that contain the abundances of 35 morphospecies, 5 substrate and microtopographic variables, and the *x*–*y* Cartesian coordinates of the 70 cores. The environmental variables are listed in Table 1.2.

1.6 A Quick Reminder about Help Sources

The **R** language is intended as a self-learning tool. So please use and abuse of the various ways to ask questions, display code, run examples that are imbedded in the framework. Some important help tools are presented in Table 1.3.

1.7 Now It Is Time...

... to get you hands full of code, numerical outputs and plots. Revise the basics of the methods, explore the code, analyse it, change it, try to apply it to your data and interpret your results. Above all, we hope to show that doing numerical ecology in **R** is fun!

Chapter 2
Exploratory Data Analysis

2.1 Objectives

Nowadays, most ecological research is done with hypothesis testing and modelling in mind. However, Exploratory Data Analysis (EDA), which uses visualization tools and computes synthetic descriptors, is still required at the beginning of the statistical analysis of multidimensional data, in order to:

- Get an overview of the data
- Transform or recode some variables
- Orient further analyses

As a worked example, we explore a classical dataset to introduce some techniques of EDA using **R** functions found in standard packages. In this chapter, you will:

- Learn or revise some bases of the **R** language
- Learn some EDA techniques applied to multidimensional ecological data
- Explore the Doubs dataset in hydrobiology as a first worked example

2.2 Data Exploration

2.2.1 Data Extraction

The Doubs dataset used here is available in the form of three comma separated values (CSV) files along with the rest of the material (see Chap. 1).

```
# Import the data from CSV files
# ******************************

# Species (community) data frame (fish abundances)
spe <- read.csv("DoubsSpe.csv", row.names=1)
```

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_2, © Springer Science+Business Media, LLC 2011

```
# Environmental data frame
env <- read.csv("DoubsEnv.csv", row.names=1)

# Spatial data frame
spa <- read.csv("DoubsSpa.csv", row.names=1)
```

Hints *At the beginning of a session, make sure to place all necessary data files and scripts in a single folder and define this folder as your working directory, either through the menu or by using function* **`setwd()`**.

If you are uncertain of the class of an object, type `class(object_name)`.

2.2.2 Species Data: First Contact

We can start data exploration, which first focuses on the community data (object `spe` created above). Verneaux used a semi-quantitative, species-specific, abundance scale (0–5) so that comparisons between species abundances make sense. However, species-specific codes cannot be understood as unbiased estimates of the true abundances (number or density of individuals) or biomasses at the sites.

We first apply some basic **R** functions and draw a barplot (Fig. 2.1):

```
# Basic functions
# ***************

spe                          # Display the whole data frame in
 # the console. Not recommended for
 # large datasets!
spe[1:5,1:10]                # Display only 5 lines and 10
columns
head(spe)                    # Display only the first few lines
nrow(spe)                    # Number of rows (sites)
ncol(spe)                    # Number of columns (species)
dim(spe)                     # Dimensions of the data frame
(rows, columns)
colnames(spe)                # Column labels (descriptors =
species)
rownames(spe)                # Row labels (objects = sites)
summary(spe)                 # Descriptive statistics for columns

 # Compare median and mean abundances. Are most distributions
 # symmetrical?

# Overall distribution of abundances (dominance codes)
# ***************************************************
```

```
# Minimum and maximum of abundance values in the whole data set
range(spe)
# Count cases for each abundance class
ab <- table(unlist(spe))
ab
# Barplot of the distribution, all species confounded
barplot(ab, las=1, xlab="Abundance class", ylab="Frequency",
  col=gray(5:0/5))
# Number of absences
sum(spe == 0)
# Proportion of zeros in the community data set
sum(spe == 0)/(nrow (spe)*ncol (spe))

  # Look at the barplot of abundance classes. How do you
  # interpret the high frequency of zeros (absences) in the
  # data frame?
```

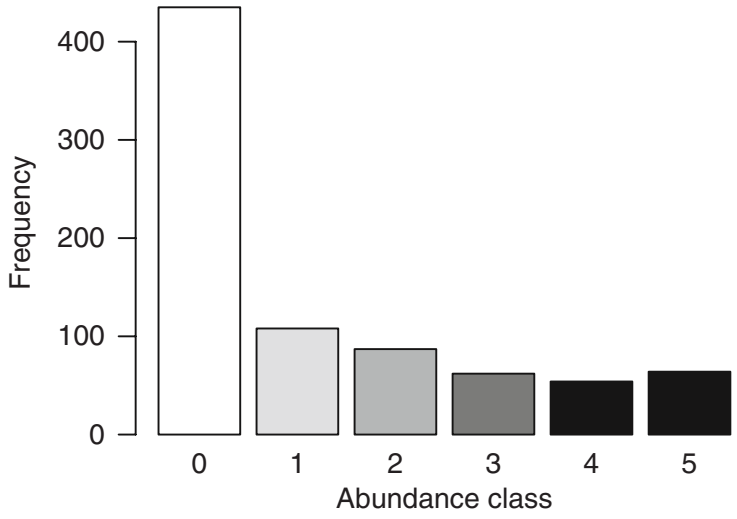

Fig. 2.1 Barplot of abundance classes

2.2.3 *Species Data: A Closer Look*

The commands above give an idea about the data structure. But codes and numbers are not very attractive or inspiring, so let us illustrate some features. We first create a map of the sites (Fig. 2.2):

```
# Map of the positions of the sites
# ********************************

# Create an empty frame (proportional axes 1:1, with titles)
# Geographic coordinates x and y from the spa data frame
plot(spa, asp=1, type="n", main="Site Locations",
  xlab="x coordinate (km)", ylab="y coordinate (km)")

# Add a blue line connecting the sites (Doubs river)
lines(spa, col="light blue")

# Add site labels
text(spa, row.names(spa), cex=0.8, col="red")

# Add text blocks
text(50, 10, "Upstream", cex=1.2, col="red")
text(30, 120, "Downstream", cex=1.2, col="red")
```

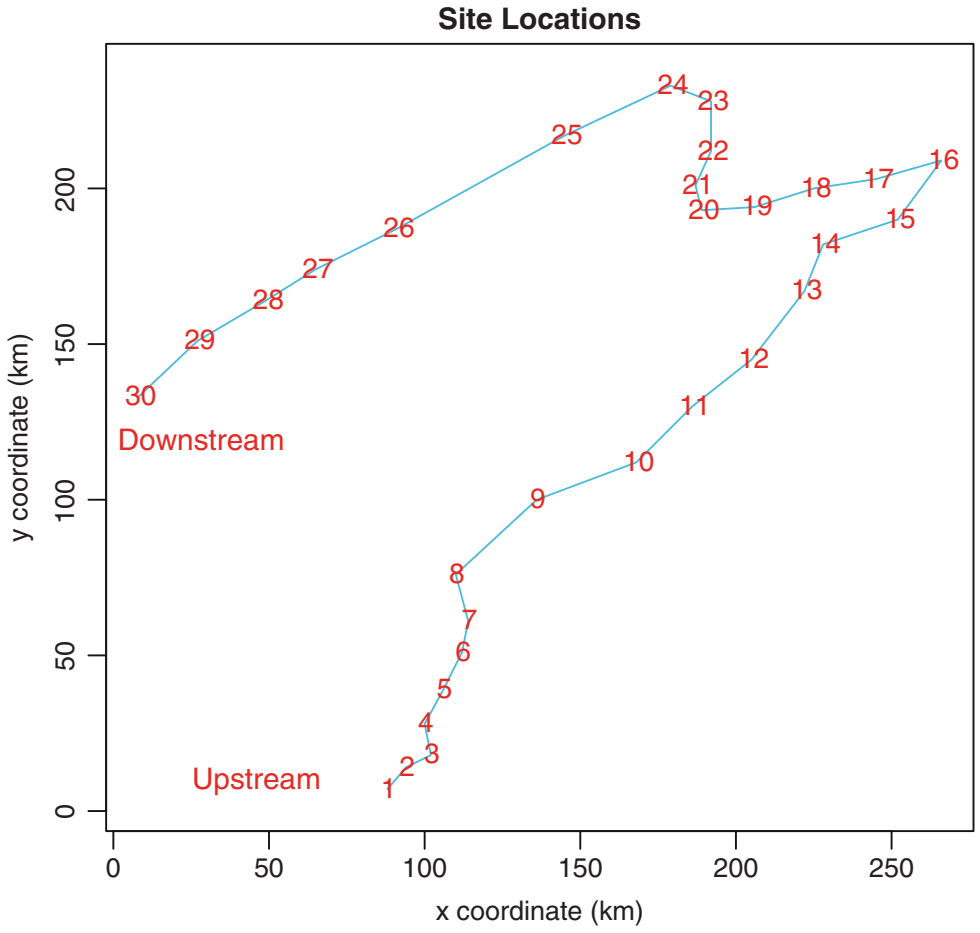

Fig. 2.2 Map of the 30 sampling sites along the Doubs River

Now, the river looks more real, but where are the fish? To show the distribution and abundance of the four species used to characterize ecological zones in European rivers (Fig. 2.3), one can type:

```
# Maps of some fish species
# ************************

# Divide the plot window into 4 frames, 2 per row
par(mfrow=c(2,2))

plot(spa, asp=1, col="brown", cex=spe$TRU, main="Brown trout",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")

plot(spa, asp=1, col="brown", cex=spe$OMB, main="Grayling",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines (spa, col="light blue")

plot(spa, asp=1, col="brown", cex=spe$BAR, main="Barbel",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")

plot(spa, asp=1, col="brown", cex=spe$BCO, main="Common bream",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")

 # From these graphs you should understand why Verneaux chose
 # these four species as ecological indicators. More about the
 # environmental conditions later…
```

Hint *Note the use of the* `cex` *argument in the* **`plot()`** *function:* `cex` *is used to define the size of an item in a graph. Here its value is a vector of the* `spe` *data frame, i.e. the abundances of a given species (e.g.* `cex=spe$TRU`*). The result is a series of bubbles whose diameter at each site is proportional to the species abundance. Also, since the object* `spa` *contains only two variables* `x` *and* `y`*, the formula has been simplified by replacing the two first arguments for horizontal and vertical axes by the name of the data frame.*

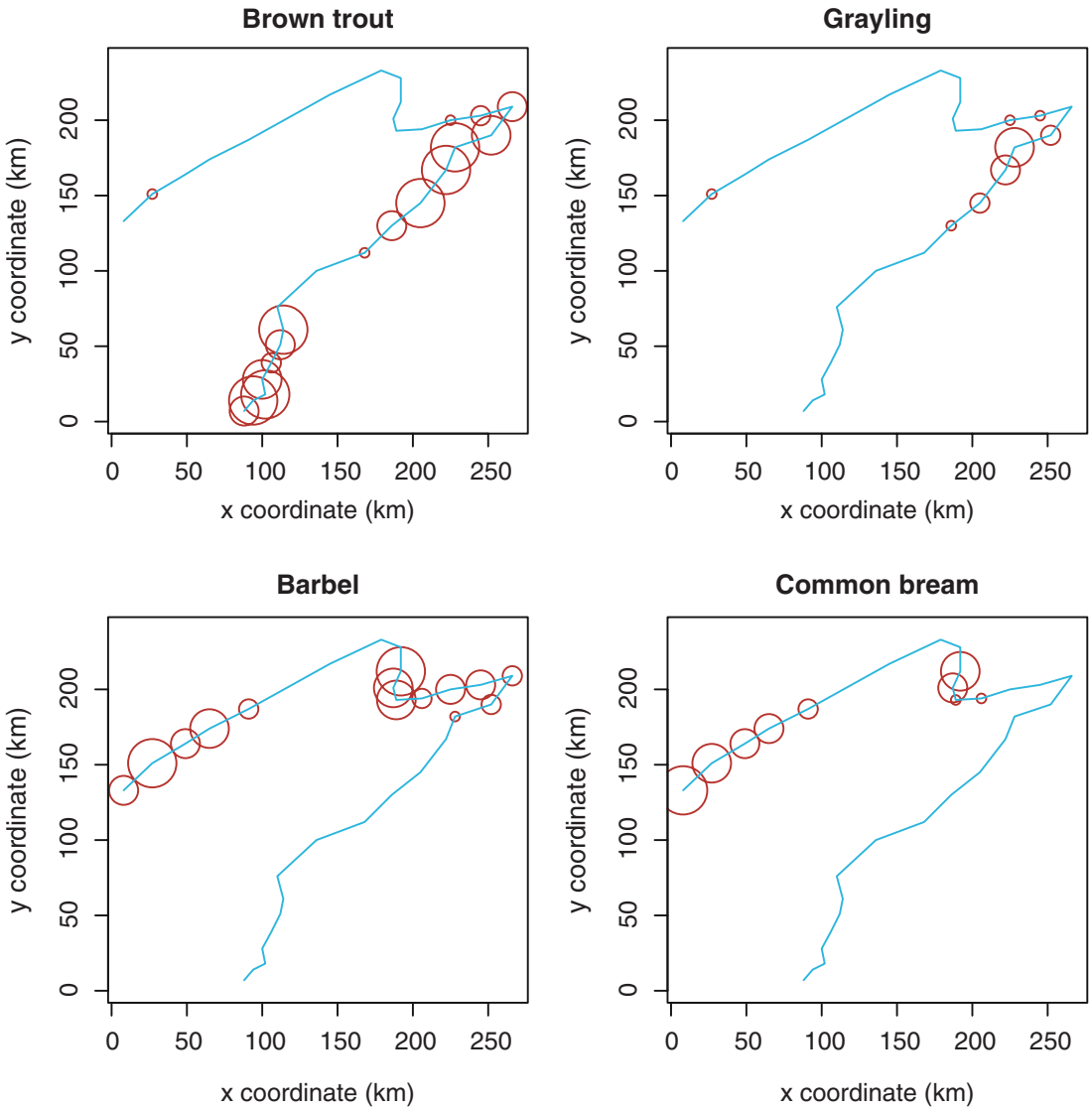

Fig. 2.3 Bubble maps of the abundance of four fish species

At how many sites does each species occur? Calculate the relative frequencies of species (proportion of the number of sites) and plot histograms (Fig. 2.4):

```
# Compare species: number of occurrences
# ************************************

# Compute the number of sites where each species is present
# To sum by columns, the second argument of apply(),
# MARGIN, is set to 2
spe.pres <- apply(spe > 0, 2, sum)
# Sort the results in increasing order
```

```
sort(spe.pres)
# Compute percentage frequencies
spe.relf <- 100*spe.pres/nrow(spe)
round(sort(spe.relf), 1)   # Round the sorted output to 1 digit

# Plot the histograms
par(mfrow=c(1,2))          # Divide the window horizontally

hist(spe.pres, main="Species Occurrences", right=FALSE, las=1,
 xlab="Number of occurrences", ylab="Number of species",
 breaks=seq(0,30,by=5), col="bisque")

hist(spe.relf, main="Species Relative Frequencies", right=FALSE,
 las=1, xlab="Frequency of occurrences (%)", ylab="Number of
 species", breaks=seq(0, 100, by=10), col="bisque")
```

Hint *Examine the use of the* **`apply()`** *function, applied here to the columns of the data frame* `spe`*. Note that the first part of the function call* (`spe > 0`) *evaluates the values in the data frame to* `TRUE`/`FALSE`*, and the number of* `TRUE` *cases per column is counted by summing.*

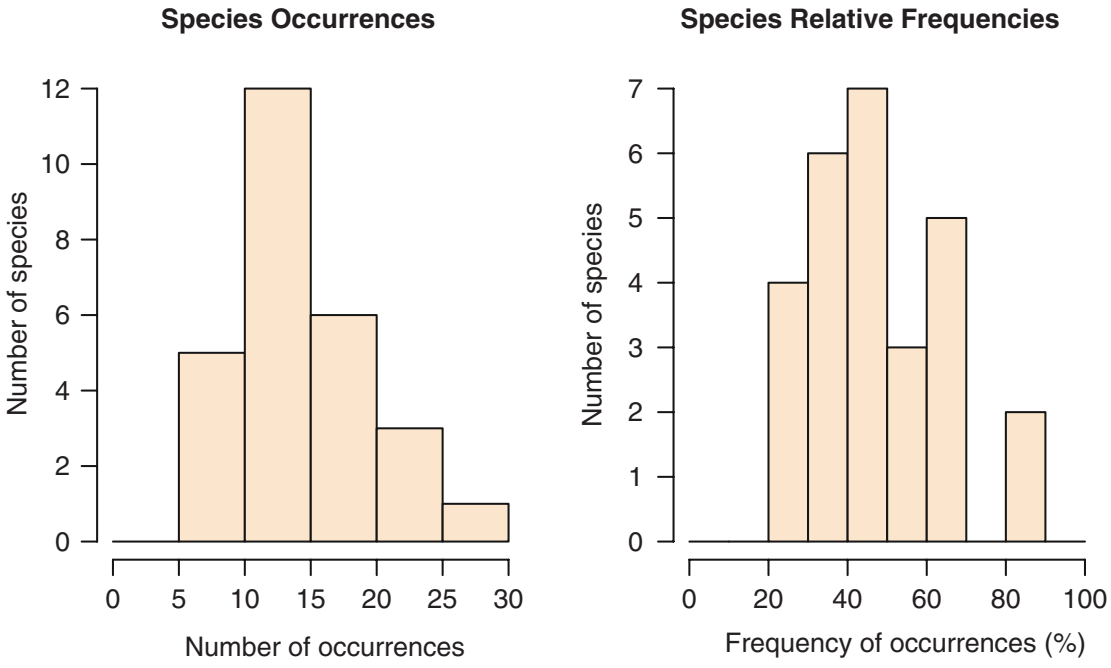

Fig. 2.4 Frequency histograms: species occurrences and relative frequencies in the 30 sites

Now that we have seen at how many sites each species is present, we may want to know how many species are present at each site (species richness, Fig. 2.5):

```
# Compare sites: species richness
# ******************************

# Compute the number of species at each site
# To sum by rows, the second argument of apply(), MARGIN,
# is set to 1
sit.pres <- apply(spe > 0, 1, sum)

# Sort the results in increasing order
sort(sit.pres)

par(mfrow=c(1,2))   # Divide the window horizontally

# Plot species richness vs. position of the sites along the
# river
plot(sit.pres, type="s", las=1, col="gray",main="Species
 Richness vs. \n Upstream-Downstream Gradient", xlab="Positions
 of sites along the river", ylab="Species richness")
text(sit.pres, row.names(spe), cex=.8, col="red")

# Use geographic coordinates to plot a bubble map
plot(spa, asp=1, main="Map of Species Richness", pch=21,
 col="white", bg="brown", cex=5*sit.pres/max(sit.pres),
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")

 # Can you identify richness hot spots along the river?
```

Hint *Observe the use of the* `type="s"` *argument of the* **`plot()`** *function to draw steps between values.*

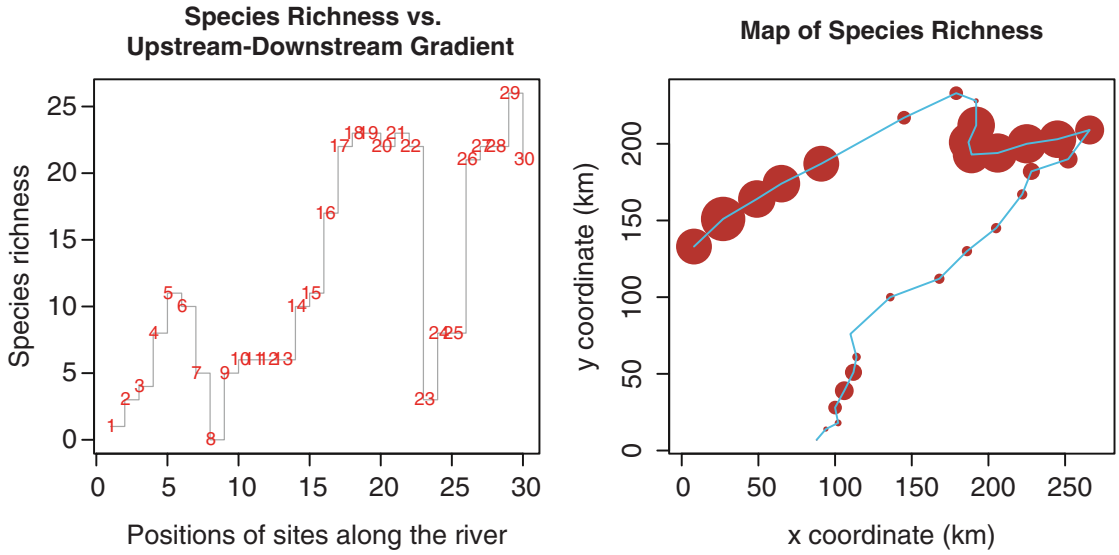

Fig. 2.5 Species richness along the river

Finally, one can easily compute classical diversity indices from the data. Let us do it with the function **`diversity()`** of the **`vegan`** package.

```
# Compute diversity indices
# ************************

# Load the required package vegan
library(vegan) # If not already loaded
# Get help on the diversity() function
?diversity

N0 <- rowSums(spe > 0)          # Species richness
H <- diversity(spe)             # Shannon entropy
N1 <- exp(H)                    # Shannon diversity number
N2 <- diversity(spe, "inv")     # Simpson diversity number
J <- H/log(N0)                  # Pielou evenness
E1 <- N1/N0                     # Shannon evenness (Hill's ratio)
E2 <- N2/N0                     # Simpson evenness (Hill's ratio)
div <- data.frame(N0, H, N1, N2, E1, E2, J)
div
```

Hint *Note the special use of function* **`rowSums()`** *for the computation of species richness N0. Normally,* `rowSums(array)` *computes the* ***sums*** *of the rows in that array. Here, argument* `spe > 0` *calls for the sum of the* ***cases*** *where the value is greater than 0.*

Hill's numbers (*N*), which are all expressed in the same units, and ratios (*E*) derived from these numbers, can be used to compute diversity indices instead of popular formulae for Shannon entropy (*H*) and Pielou evenness (*J*). Note that there are other ways of estimating diversity while taking into account unobserved species (e.g. Chao and Shen 2003).

2.2.4 Species Data Transformation

There are instances where one needs to transform the data prior to analysis. The main reasons are given below with examples of transformations:

- Make descriptors that have been measured in different units comparable (ranging, standardization to *z*-scores, i.e. centring and reduction, also called scaling)
- Make the variables normal and stabilize their variances (e.g. square root, fourth root, log transformations)
- Make the relationships among variables linear (e.g. log transformation of response variable if the relationship is exponential)
- Modify the weights of the variables or objects (e.g. give the same length (or norm) to all object vectors)
- Code categorical variables into dummy binary variables or Helmert contrasts

Species abundances are dimensionally homogenous (expressed in the same physical units), quantitative (count, density, cover, biovolume, biomass, frequency, etc.) or semi-quantitative (classes) variables and restricted to positive or null values (zero meaning absence). For these, *simple transformations* may be used to reduce the importance of observations with very high values: `sqrt()` (square root), `sqrt(sqrt())` (fourth root), or `log1p()` (natural logarithm of abundance + 1 to keep absence as zero) are commonly applied **R** functions. In extreme cases, to give the same weight to all positive abundances irrespective of their values, the data can be transformed to binary 1-0 form (presence–absence).

The `decostand()` function of the `vegan` package provides many options for common standardization of ecological data. In this function, *standardization*, as contrasted with simple transformation (such as square root, log or presence–absence), means that the values are not transformed individually but relative to other values in the data table. Standardization can be done relative to sites (site profiles), species (species profiles), or both (double profiles), depending on the focus of the analysis. Here are some examples illustrated by boxplots (Fig. 2.6):

```
# Data transformation and standardization
########################################

# Get help on the decostand() function
?decostand

# Simple transformations
# **********************

# Partial view of the raw data (abundance codes)
spe[1:5,2:4]

# Transform abundances to presence-absence (1-0)
spe.pa <- decostand(spe, method="pa")
spe.pa[1:5,2:4]

# Species profiles: 2 methods;
# presence-absence or abundance data
 ***********************************

# Scale abundances by dividing them by the maximum value for
# each species
# Note: MARGIN=2 (default value) for this method
spe.scal <- decostand(spe, "max")
spe.scal[1:5,2:4]
# Display the maximum by column
apply(spe.scal, 2, max)

  # Did the scaling work properly? It is good to keep an eye on
  # your results by a plot or by the use of summary statistics.

# Scale abundances by dividing them by the species totals
# (relative abundance per species)
# Note: MARGIN=2 for this method
spe.relsp <- decostand(spe, "total", MARGIN=2)
spe.relsp[1:5,2:4]
# Display the sum by column
apply(spe.relsp,2,sum)

# Site profiles: 3 methods; presence-absence or abundance data
# ************************************************************

# Scale abundances by dividing them by the site totals
# (relative abundances, or relative frequencies, per site)
# Note: MARGIN=1 (default value) for this method
spe.rel <- decostand(spe, "total")      # default MARGIN = 1
```

```
spe.rel[1:5,2:4]
# Display the sum of row vectors to determine if the scaling
# worked properly
apply(spe.rel, 1, sum)

# Give a length of 1 to each row vector (Euclidean norm)
spe.norm <- decostand(spe, "normalize")
spe.norm[1:5,2:4]
# Verify the norm of row vectors
norm <- function(x) sqrt(x%*%x)
apply(spe.norm, 1, norm)

 # The scaling above is called the 'chord transformation': the
 # Euclidean distance function applied to chord-transformed
 # data produces a chord distance matrix (Chapter 3). Useful
 # before PCA and RDA (Chapters 5 and 6) and k-means
 # partitioning (Chapter 4).

# Compute relative frequencies by rows (site profiles),
# then square root
spe.hel <- decostand(spe, "hellinger")
spe.hel[1:5,2:4]
# Check the norm of row vectors
apply(spe.hel,1,norm)

 # This is called the Hellinger transformation. The Euclidean
 # distance function applied to Hellinger-transformed data
 # produces a Hellinger distance matrix (Chapter 3). Useful
 # before PCA and RDA (Chapters 5 and 6) and k-means
 # partitioning (Chapter 4).
 # Note: the Hellinger transformation can also be obtained by
 # applying the chord transformation to square-root-
 # transformed species data.

# Standardization of both species and sites (double profiles)
# ************************************************************

# Chi-square transformation
spe.chi <- decostand(spe, "chi.square")
spe.chi[1:5,2:4]
# Check what happened to site 8 where no species was found
spe.chi[7:9,]

 # The Euclidean distance function applied to chi-square-
 # transformed data produces a chi-square distance matrix
 # (Chapter 3).
```

```
# Wisconsin standardization: abundances are first ranged by
# species maxima and then by site totals
spe.wis <- wisconsin(spe)
spe.wis[1:5,2:4]

# Boxplots of transformed abundances of a common species
# (stone loach)
# ****************************************************

par(mfrow=c(2,2))

boxplot(spe$LOC, sqrt(spe$LOC), log1p(spe$LOC),
 las=1, main="Simple transformation",
 names=c("raw data", "sqrt", "log"), col="bisque")

boxplot(spe.scal$LOC, spe.relsp$LOC,
 las=1, main="Standardization by species",
 names=c("max", "total"), col="lightgreen")

boxplot(spe.hel$LOC, spe.rel$LOC, spe.norm$LOC,
 las=1, main="Standardization by sites",
 names=c("Hellinger", "total", "norm"), col="lightblue")

boxplot(spe.chi$LOC, spe.wis$LOC,
 las=1, main="Double standardization",
 names=c("Chi-square", "Wisconsin"), col="orange")

 # Compare the effect of these transformations and standardiza-
 # tions on the range and distribution of the scaled abundances.
```

Hint *Take a look at the line:* `norm <- function(x) sqrt(x%*%x)`. *It is an example of a small function built on the fly to fill a gap in the standard* **R** *packages: this function computes the norm (length) of a vector using a matrix algebraic form of Pythagora's theorem. For more matrix algebra, visit the* ***Code It Yourself*** *corners.*

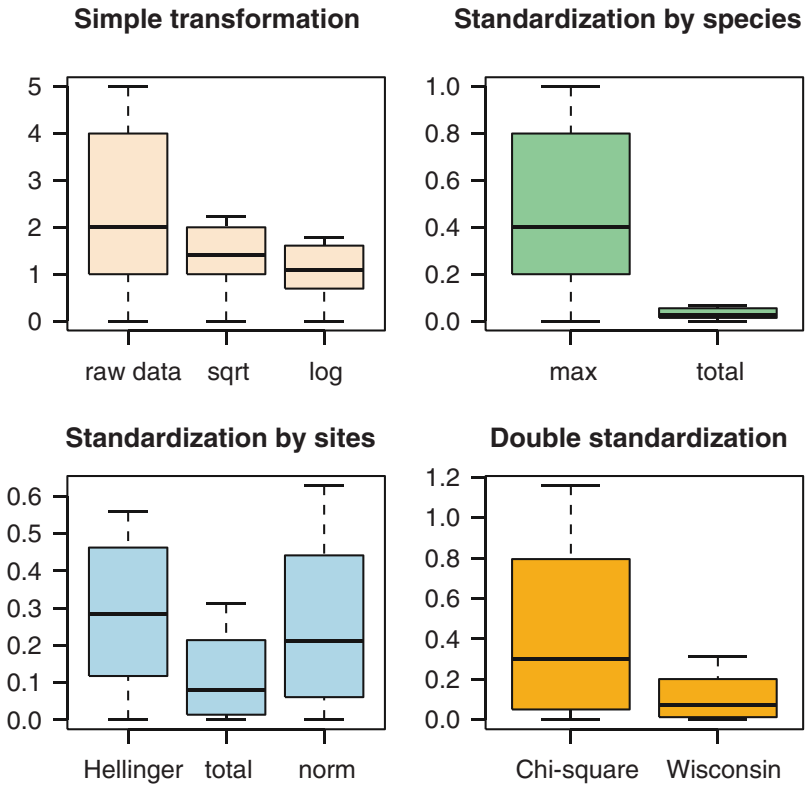

Fig. 2.6 Boxplots of transformed abundances of a common species, *Nemacheilus barbatulus* (stone loach)

Another way to compare the effects of transformations on species profiles is to plot them along the river course:

```
# Plot profiles along the upstream-downstream gradient
# ***************************************************

par(mfrow=c(2,2))
plot(env$das, spe$TRU, type="l", col=4, main="Raw data",
  xlab="Distance from the source [km]", ylab="Raw abundance
  code")
lines(env$das, spe$OMB, col=3)
lines(env$das, spe$BAR, col="orange")
lines(env$das, spe$BCO, col=2)
lines(env$das, spe$LOC, col=1, lty="dotted")
```

```
plot(env$das, spe.scal$TRU, type="l", col=4, main="Species
 profiles (max)",xlab="Distance from the source [km]",
 ylab="Standardized abundance")
lines(env$das, spe.scal$OMB, col=3)
lines(env$das, spe.scal$BAR, col="orange")
lines(env$das, spe.scal$BCO, col=2)
lines(env$das, spe.scal$LOC, col=1, lty="dotted")

plot(env$das, spe.hel$TRU, type="l", col=4, main="Site profiles
 (Hellinger)", xlab="Distance from the source [km]",
 ylab="Standardized abundance")
lines(env$das, spe.hel$OMB, col=3)
lines(env$das, spe.hel$BAR, col="orange")
lines(env$das, spe.hel$BCO, col=2)
lines(env$das, spe.hel$LOC, col=1, lty="dotted")

plot(env$das, spe.chi$TRU, type="l", col=4, main="Double
 profiles (Chi-square)", xlab="Distance from the source [km]",
 ylab="Standardized abundance")
lines(env$das, spe.chi$OMB, col=3)
lines(env$das, spe.chi$BAR, col="orange")
lines(env$das, spe.chi$BCO, col=2)
lines(env$das, spe.chi$LOC, col=1, lty="dotted")
legend("topright", c("Brown trout", "Grayling", "Barbel",
 "Common bream", "Stone loach"), col=c(4,3,"orange",2,1),
 lty=c(rep(1,4),3))

  # Compare the profiles and explain the differences.
```

The Code It Yourself corner #1

Write a function to compute the Shannon–Weaver entropy for a site vector containing species abundances. The formula is:

$$H' = -\sum [p_i \times \log(p_i)]$$

where $p_i = n_i / N$ *and* n_i *= abundance of species i and N = total abundance of all species.*

After that, display the code of **vegan***'s function* **diversity()** *to see how it has been coded among other indices by Jari Oksanen and Bob O'Hara. Nice and compact, isn't it?*

2.2.5 Environmental Data

Now that we are acquainted with the species data, let us turn to the environmental data (object `env`).

First, go back to Sect. 2.2.2 and apply the basic functions presented there to `env`. While examining the **`summary()`**, note how the variables differ from the species data in values and spatial distributions.

Draw maps of some of the environmental variables, first in the form of bubble maps (Fig. 2.7):

```
# Bubble maps of some environmental variables
# ******************************************

par(mfrow=c(2,2))

plot(spa, asp=1, main="Altitude", pch=21, col="white", bg="red",
  cex=5*env$alt/max(env$alt), xlab="x", ylab="y")
lines(spa, col="light blue")

plot(spa, asp=1, main="Discharge", pch=21, col="white",
  bg="blue", cex=5*env$deb/max(env$deb), xlab="x", ylab="y")
lines(spa, col="light blue")

plot(spa, asp=1, main="Oxygen", pch=21, col="white",
  bg="green3", cex=5*env$oxy/max(env$oxy), xlab="x", ylab="y")
lines(spa, col="light blue")

plot(spa, asp=1, main="Nitrate", pch=21, col="white",
  bg="brown", cex=5*env$nit/max(env$nit), xlab="x", ylab="y")
lines(spa, col="light blue")

  # Which ones of these maps display an upstream-downstream
  # gradient? How could you explain the spatial patterns of the
  # other variables?
```

Hint *See how the* `cex` *argument is used to make the size of the bubbles comparable among plots. Play with these values to see the changes in the graphical output.*

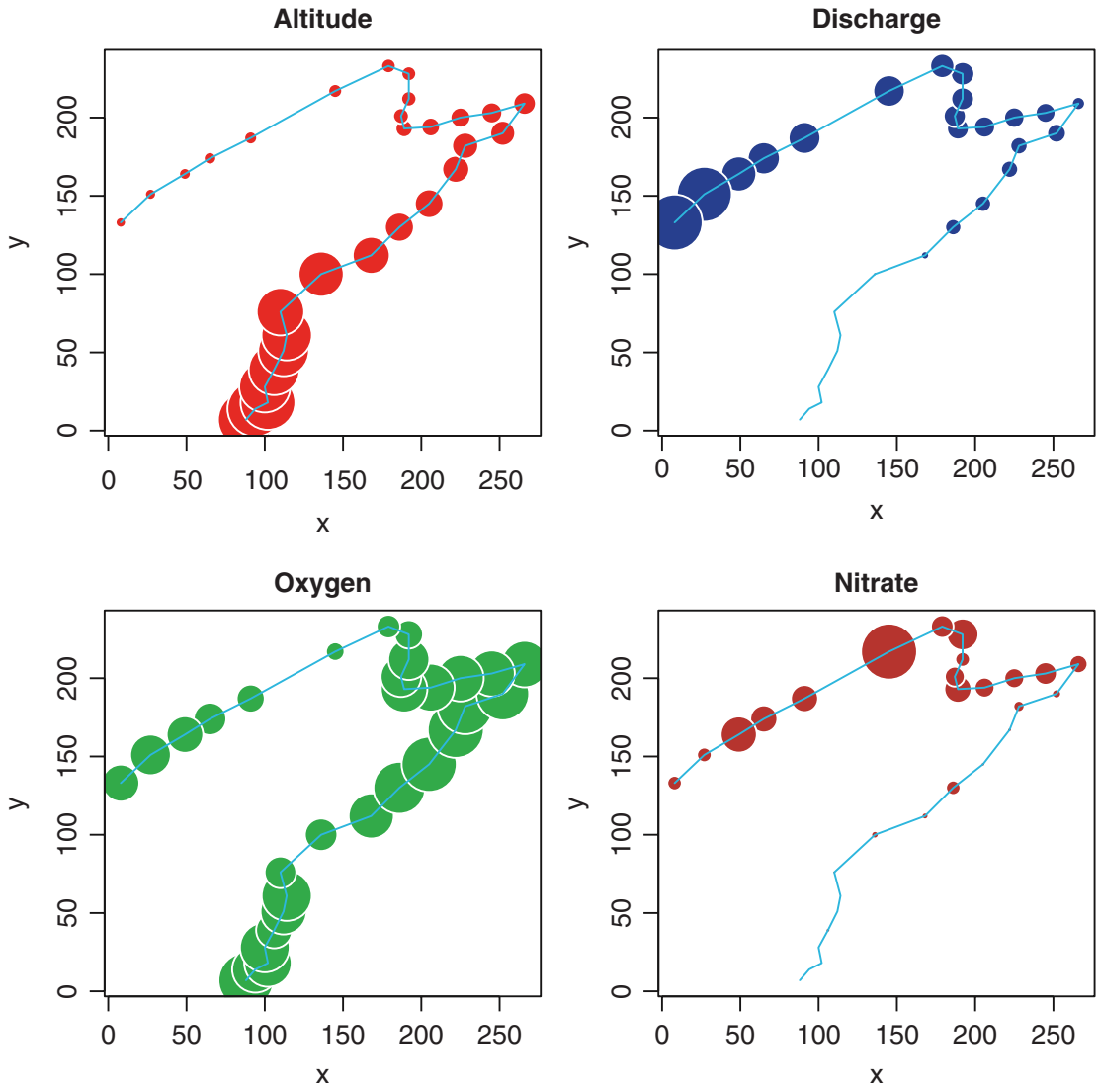

Fig. 2.7 Bubble maps of environmental variables

Now, examine the variation of some descriptors along the stream (Fig. 2.8):

```
# Line plots
# **********

par(mfrow=c(2,2))
plot(env$das, env$alt, type="l", xlab="Distance from the source
  (km)", ylab="Altitude (m)", col="red", main="Altitude")
plot(env$das, env$deb, type="l", xlab="Distance from the source
  (km)", ylab="Discharge (m3/s)", col="blue", main="Discharge")
```

```
plot(env$das, env$oxy, type="l", xlab="Distance from the source
  (km)", ylab="Oxygen (mg/L)", col="green3", main="Oxygen")
plot(env$das, env$nit, type="l", xlab="Distance from the source
  (km)", ylab="Nitrate (mg/L)", col="brown", main="Nitrate")
```

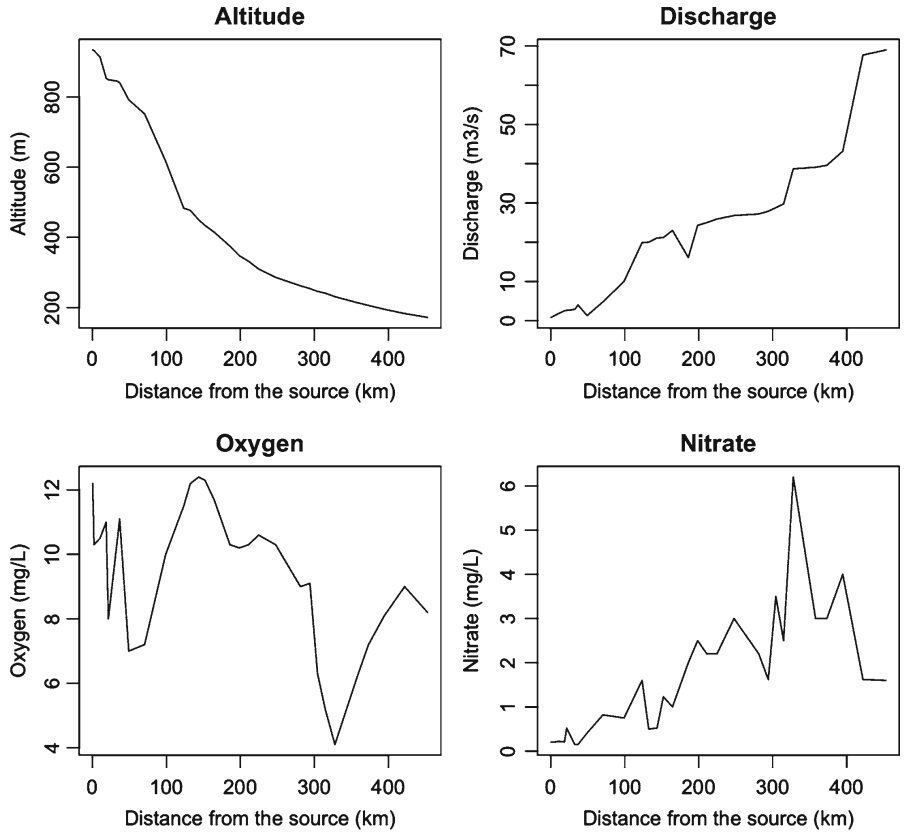

Fig. 2.8 Line plots of environmental variables

To explore graphically the bivariate relationships among the environmental variables, we can use the powerful **`pairs()`** graphical function, which draws a matrix of scatter plots (Fig. 2.9).

Moreover, we can add a LOWESS smoother to each bivariate plot and draw histograms in the diagonal plots, showing the frequency distribution of each variable, using external functions of the **panelutils.R** script.

```
# Scatter plots for all pairs of environmental variables
# ******************************************************

# Load additional functions from an R script
source("panelutils.R")     # panelutils.R must be in the working
                           # directory

# Bivariate plots with histograms on the diagonal and smooth
# fitted curves
op <- par(mfrow=c(1,1), pty="s")
pairs(env, panel=panel.smooth, diag.panel=panel.hist,
 main="Bivariate Plots with Histograms and Smooth Curves")
par(op)

 # From the histograms, do many variables seem normally
 # distributed?
 # Note that normality is not required for explanatory variables
 # in regression analysis and in canonical ordination.
 # Do many scatter plots show linear or at least monotonic
 # relationships?
```

Hint *Each scatterplot shows the relationship between two variables identified on the diagonal. The abscissa of the scatterplot is the variable above or under it, and the ordinate is the variable to its left or right.*

Simple transformations, such as the log transformation, can be used to improve the distributions of some variables (make it closer to the normal distribution). Furthermore, because environmental variables are dimensionally heterogeneous (expressed in different units and scales), many statistical analyses require their standardization to zero mean and unit variance. These centred and scaled variables are called z-scores. We can now illustrate transformations and standardization with our example data (Fig. 2.10).

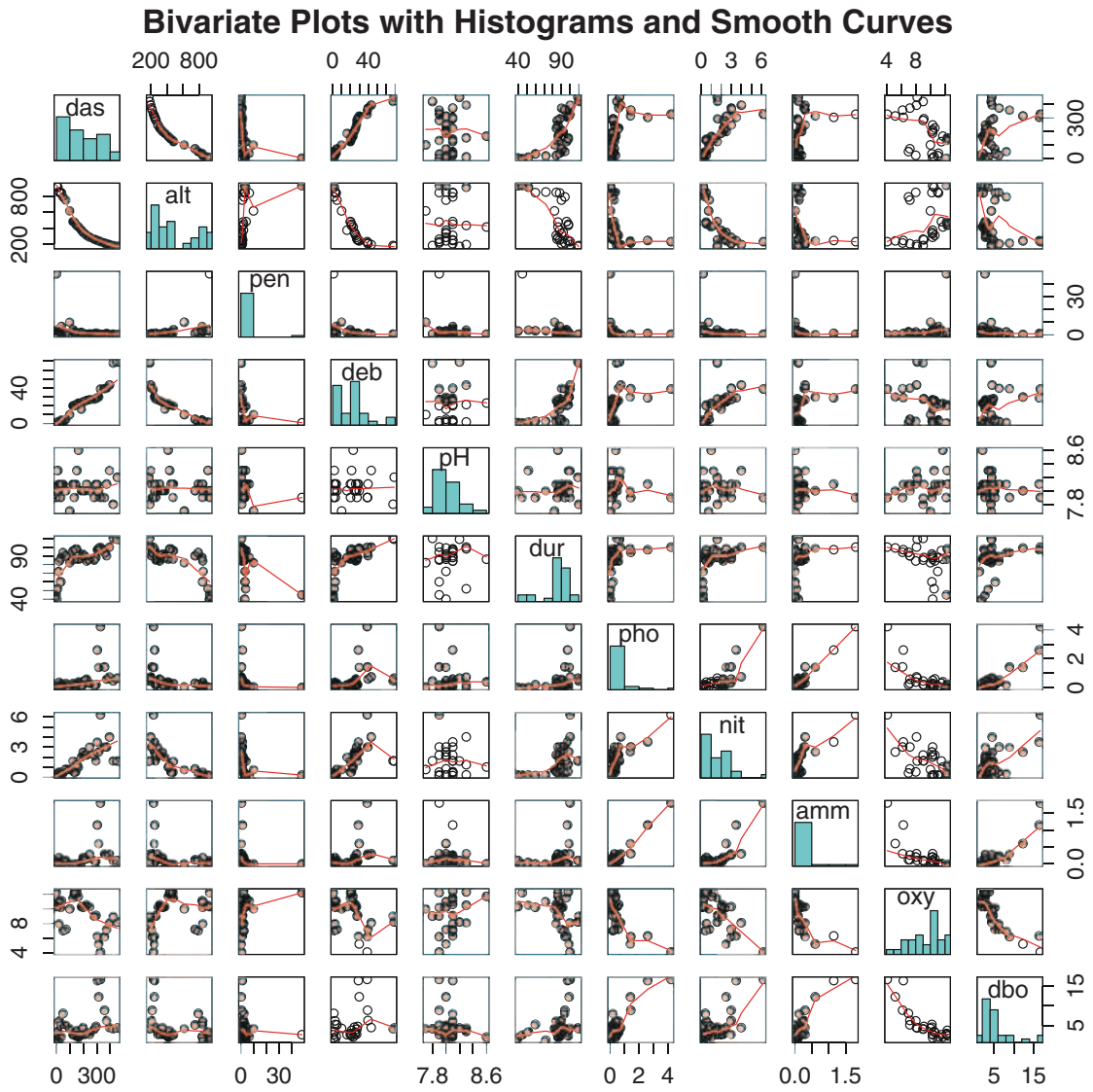

Fig. 2.9 Scatter plots between all pairs of environmental variables with LOWESS smoothers

```
# Simple transformation of an environmental variable
# ************************************************

range(env$pen)
# Log-transformation of the variable 'slope' (y = ln(x))
# Compare histograms and boxplots of raw and transformed values
par(mfrow=c(2,2))
```

```
hist(env$pen, col="bisque", right=F)
hist(log(env$pen), col="light green", right=FALSE,
main="Histogram of ln(env$pen)")
boxplot(env$pen, col="bisque", main="Boxplot of env$pen",
 ylab="env$pen")
boxplot(log(env$pen), col="light green", main="Boxplot of
 ln(env$pen)", ylab="log(env$pen)")

# Standardization of all environmental variables
# *********************************************

# Center and scale = standardize variables (z-scores)
env.z <- decostand(env, "standardize")
apply(env.z, 2, mean)      # means = 0
apply(env.z, 2, sd)        # standard deviations = 1

# Same standardization using the scale() function (which returns
# a matrix)
env.z <- as.data.frame(scale(env))
```

Hint *Normality of a vector can be tested by using the Shapiro–Wilk test, available through function* **`shapiro.test()`**.

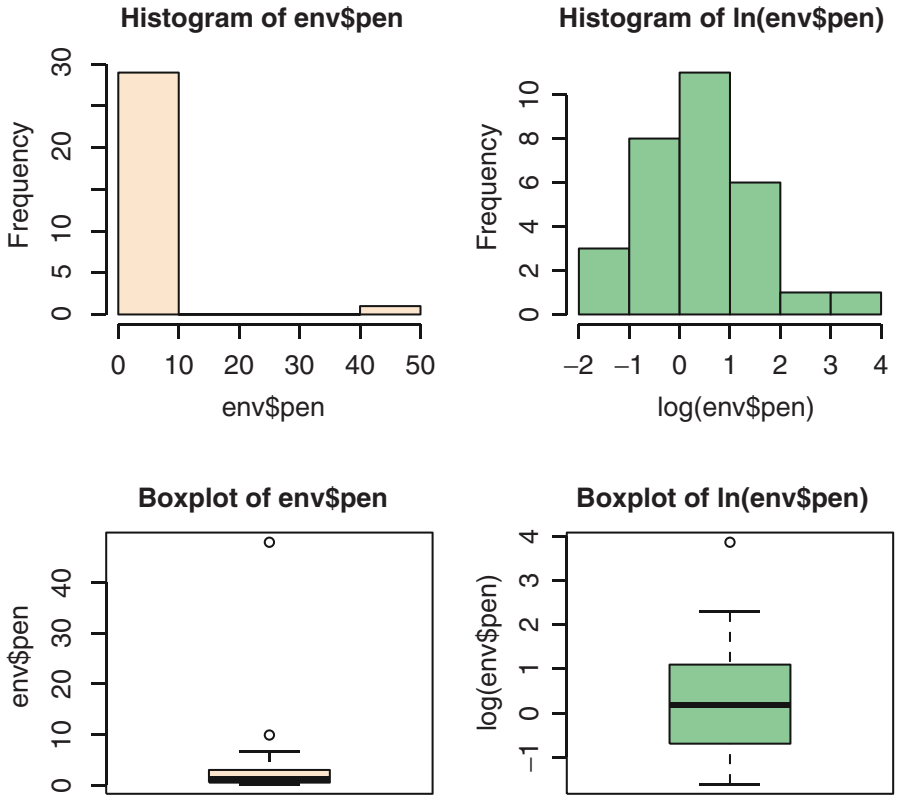

Fig. 2.10 Histograms and boxplots of the untransformed (*left*) and log-transformed `pen` variable (slope)

2.3 Conclusion

The tools presented in this chapter allow researchers to obtain a general impression of their data. Although you see much more elaborate analyses in the next chapters, keep in mind that a first exploratory look at the data can tell much about them. Information about simple parameters and distributions of variables is important to consider in order to choose more advanced analyses correctly. Graphical representations like bubble maps are useful to reveal how the variables are spatially organized; they may help generate hypotheses about the processes acting behind the scene. Boxplots and simple statistics may be necessary to reveal unusual or aberrant values.

EDA is often neglected by people who are eager to jump to more sophisticated analyses. We hope to have convinced you that it should have an important place in the toolbox of ecologists.

Chapter 3
Association Measures and Matrices

3.1 Objectives

Most methods of multivariate analysis, in particular ordination and clustering techniques, are explicitly or implicitly[1] based on the comparison of all possible pairs of objects or descriptors. The comparisons take the form of association measures (often called coefficients or indices), which are assembled in a square and symmetrical association matrix, of dimensions $n \times n$ when objects are compared, or $p \times p$ when variables are compared. Since the subsequent analyses are done on the association matrix, the choice of an appropriate measure is crucial. In this chapter, you will:

- Quickly revise the main categories of association coefficients
- Learn how to compute, examine and visually compare dissimilarity matrices (Q mode) and dependence matrices (R mode)
- Apply these techniques to a classical dataset
- Learn or revise some basics of programming functions with the **R** language

3.2 The Main Categories of Association Measures (Short Overview)

It is beyond the scope of this book to explain the various methods in detail, but it is useful to provide a wrap-up of the main categories of measures. This facilitates the choice of an appropriate index in many situations, and improves the understanding of the applications proposed below. Note that we use the expressions "measure",

[1]The association measure among objects may be implicit. It is the Euclidean distance in principal component analysis (PCA, Chap. 5) and *k*-means partitioning (Chap. 4), for example, and the chi-square distance in correspondence analysis (CA, Chap. 5).

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_3, © Springer Science+Business Media, LLC 2011

"index" and "coefficient" as synonyms to refer to the quantities used to compare pairs of objects or variables.[2]

3.2.1 Q Mode and R Mode

When pairs of objects are compared, the analysis is said to be in the *Q mode*. When pairs of descriptors are compared, the analysis is said to be in the *R mode*. This distinction is important because the association measures in Q- and R-mode analyses are not the same.

In *Q mode*, the association measure is the *distance* (dissimilarity) or the *similarity* between pairs of objects, e.g. Euclidean distance, Jaccard similarity. In *R mode*, one uses a measure of *dependence* among variables, such as the covariance or correlation coefficient.

3.2.2 Symmetrical or Asymmetrical Coefficients in Q Mode: The Double-Zero Problem

Virtually, all distance or similarity measures used in ecology are *symmetric* in one sense: the value of the coefficient between objects n_1 and n_2 is the same as the value of the coefficient between objects n_2 and n_1. The same holds for dependence measures in the R mode. The problem addressed here is different. It concerns the treatment of double-zeros in the comparison of pairs of objects.

In certain cases, the zero value has the same meaning as any other value along the scale of a descriptor. For instance, the absence (0 mg/L) of dissolved oxygen in the deep layers of a lake is an ecologically meaningful information: the concentration is below the measurement threshold and this poses severe constraints on aerobic life forms, whatever the reason for this condition.

On the contrary, the zero value in a matrix of species abundances (or presence–absence) is much more tricky to interpret. The presence of a species at two sites generally implies that these sites provide a similar set of minimal conditions allowing the species to survive; these conditions are the dimensions of its ecological niche. The absence of a species from a relevé or site, however, may be due to a variety of causes: the species' niche may be occupied by a replacement species, or it may not have reached that site even though the conditions are favourable, or the

[2]Although the term "coefficient" is sometimes narrowly defined as a multiplicative factor of a variable in a mathematical expression, it has been applied for decades to the broader sense used in this book.

absence of the species is due to different non-optimal conditions on *any* of the important dimensions of its ecological niche, or the species is present but has not been observed or captured by the researcher, or the species does not show a regular distribution among the sites under study. The key points here are that (1) in most situations, the absence of a given species from two sites cannot readily be counted as an indication of resemblance between these sites because this double absence may be due to completely different reasons, and (2) the number of uninterpretable double zeros in a species matrix depends on the number of species and thus increases strongly with the number of rare species detected.

The information "presence" thus has a clearer interpretation than the information "absence". One can distinguish two classes of association measures based on this problem: the coefficients that consider the double zeros (sometimes also called "negative matches") as indications of resemblance (like any other value) are said to be *symmetrical*, the others, *asymmetrical*.[3] In most cases, it is preferable to use asymmetrical coefficients when analysing species data, unless one has compelling reasons to consider each double absence in a matrix as being due to the same cause. Possible examples of such exceptions are controlled experiments with known community compositions or ecologically homogeneous areas with disturbed zones.

3.2.3 Association Measures for Qualitative or Quantitative Data

Some variables are qualitative (nominal or categorical, either binary or multiclass), others are semi-quantitative (ordinal) or quantitative (discrete or continuous). Association coefficients exist for all types of variables, but most of them fall into two classes: coefficients for binary variables (hereunder called *binary coefficients* for short, although it is the variables that are binary, not the values of the association measures) and coefficients for quantitative variables (called *quantitative coefficients* hereafter).

3.2.4 To Summarize…

Keep track on what kind of association measure you need. Before any analysis, ask the following questions:

- Are you comparing objects (Q-mode) or variables (R-mode analysis)?
- Are you dealing with species data (usually asymmetrical coefficients) or other types of variables (symmetrical coefficients)?

[3] The use of the words *symmetrical/asymmetrical* for this distinction, as opposed to *symmetric/asymmetric* (same value of the coefficient between n_1 and n_2 as between n_2 and n_1), follows Legendre and Legendre (1998).

- Are your data binary (binary coefficients), quantitative (quantitative coefficients) or mixed or of other types (e.g. ordinal; special coefficients)?

In the following sections, you will explore various possibilities. In most cases, more than one association measure is available to study a given problem.

3.3 Q Mode: Computing Distance Matrices Among Objects

In the Q mode, we use four packages: **stats** (included in the standard installation of **R**), **vegan**, **ade4**, **cluster** and **FD**. Note that this list of packages is by far not exhaustive, but it should satisfy the needs of most ecologists.

Although the literature provides similarity as well as distance measures, in **R** all similarity measures are converted to distances to compute a square matrix of class `"dist"` in which the diagonal (distance between each object and itself) is 0 and can be ignored. The conversion formula varies with the package used, and this may not be without consequences:

- In **stats**, **FD** and **vegan**, the conversion from similarities S to dissimilarities D is $D = 1 - S$.
- In **ade4**, it is computed as $D = \sqrt{1 - S}$. This allows some indices to become Euclidean, a geometrical property that is useful in some analyses, e.g., principal coordinate analysis (see Chap. 5). We come to it when it becomes relevant. Distance matrices computed by other packages that are not Euclidean can often be made Euclidean by computing `D2 <- sqrt(D)`.
- In **cluster**, all available measures are distances, so no conversion has to be made.

Therefore, although we stick to the conventional names of the most classical coefficients, as they can be found in textbooks, it is implicit from now on that all these measures have been converted to distances when computed by **R** functions. For instance, the Jaccard (1901) community index is basically a similarity, but the output of the computation of that coefficient in **stats**, **vegan** and **ade4** is a distance matrix.

3.3.1 Q Mode: Quantitative Species Data

Let us use the fish species dataset `spe` again. We consider the data as quantitative although, strictly speaking, the values do not represent raw fish numbers.

Quantitative species data generally require asymmetric distance measures. In this category, frequently used coefficients are the Bray–Curtis dissimilarity D_{14}[4] (also known as the reciprocal of the Steinhaus similarity index, S_{17}), the chord distance D_3, the chi-square distance D_{21} and the Hellinger distance D_{17}. Let us compute dissimilarity matrices using some of these indices. In the process, we use the package **gclus** for visualization purposes.

Bray–Curtis dissimilarity matrices can be computed directly from raw data, although true abundances are often log-transformed because D_{14} gives the same importance to absolute differences in abundance irrespective of the order of magnitude of the abundances. In this coefficient, a difference of five individuals has the same weight when the abundances are three and eight as when the abundances are 6203 and 6208.

The *chord* distance is a Euclidean distance computed on site vectors normalized to length 1; this normalization is called the chord transformation. The normalization is done by **vegan**'s function **decostand()**, argument `normalize`.[5]

The *Hellinger* distance is a Euclidean distance on site vectors, where the abundance values are first divided by the site total abundance, and the result is square-root transformed; this is called the Hellinger transformation. Another description of this transformation is to apply a chord transformation on square-root-transformed data. Conversely, chord-transformed abundance data are obtained by a Hellinger transformation of squared abundance data. This relationship shows the relatedness of the chord and Hellinger distances and emphasizes the effect of the Hellinger transformation to reduce the importance of large abundances. The Hellinger transformation is obtained in one step by **decostand()** with the argument `hellinger`.

```
# Load required  packages
library(ade4)
library(vegan)         # should be loaded after ade4 to avoid
                       # some conflicts
library(gclus)
library(cluster)
library(FD)
```

[4] In this book, symbols and numbering of similarity and distance measures are taken from Legendre and Legendre (1998).

[5] This way of presenting several distance measures (as pre-transformations followed by computation of a Euclidean distance) was described by Legendre and Gallagher (2001). More about this topic in Sect. 3.5 at the end of this chapter.

```
# Import the data from CSV files
spe <- read.csv("DoubsSpe.csv", row.names=1)
env <- read.csv("DoubsEnv.csv", row.names=1)
spa <- read.csv("DoubsSpa.csv", row.names=1)
# Remove empty site 8
spe <- spe[-8,]
env <- env[-8,]
spa <- spa[-8,]

# Dissimilarity and distance measures for (semi-)quantitative
# data
# ********************************************************

# Bray-Curtis dissimilarity matrix on raw species data
spe.db <- vegdist(spe)     # Bray-Curtis dissimilarity (default)
head(spe.db)
# Bray-Curtis dissimilarity matrix on log-transformed abundances
spe.dbln <- vegdist(log1p(spe))
head(spe.dbln)
# Chord distance matrix
spe.norm <- decostand(spe, "nor")
spe.dc <- dist(spe.norm)
head(spe.dc)
# Hellinger distance matrix
spe.hel <- decostand(spe, "hel")
spe.dh <- dist(spe.hel)
head(spe.dh)
 # Examine the help files of decostand() and vegdist() and find
 # how to compute a matrix of chi-square distances, which would
 # also be an appropriate choice for these data.
```

Hint *Type* `?log1p` *to see what you have done to the data prior to building the second Bray–Curtis dissimilarity matrix. Why use* **`log1p()`** *rather than* **`log()`***?*

3.3.2 Q Mode: Binary (Presence–Absence) Species Data

When the only data available are binary, or when the abundances are irrelevant, or, sometimes, when the data table contains quantitative values of uncertain or unequal quality, the analyses are done on presence–absence (1-0) data.

The Doubs fish dataset is quantitative, but for the sake of the example we recode it to binary form (using the appropriate arguments when needed): all values larger than 0 are given a value equal to 1. The exercise consists in computing several dissimilarity matrices based on appropriate similarity coefficients: the Jaccard (S_7) and Sørensen (S_8) similarities. For each pair of sites, the Jaccard similarity is the ratio

between the number of double 1s and the number of species, excluding the species represented by double zeros in the pair of objects considered. Therefore, a Jaccard similarity of 0.25 means that 25% of the total number of species observed at two sites were present in both sites and 75% in one site only. The Jaccard distance computed in **R** is either $(1-0.25)$ or $\sqrt{1-0.25}$ depending on the package used. The Sørensen similarity (S_8) gives more weight to the number of double 1s and its reciprocal (complement to 1) is equivalent to the Bray–Curtis distance computed on species presence–absence data.

A further interesting relationship is that the Ochiai similarity (S_{14}), which is also appropriate for species presence–absence data, is related to the chord and Hellinger distances. Computing either one of these distances on presence–absence data, followed by division by $\sqrt{2}$, produces $\sqrt{1-\text{Ochiai similarity}}$. This reasoning shows that the chord and Hellinger transformations are meaningful for species presence–absence data. This is also the case for the chi-square transformation: the Euclidean distance computed on data transformed in that way produces the chi-square distance, which is appropriate for both quantitative and presence–absence data.

```
# Dissimilarity measures for binary data
# ************************************
 # Note: a binary transformation of the data(decostand(..,"pa"))
 # is not necessary, since all binary distance functions make
 # the data binary before computing the coefficients.
 # dist.binary() does that automatically
 # whereas vegdist() requires an argument binary=TRUE

# Jaccard dissimilarity matrix using function vegdist()
spe.dj <- vegdist(spe, "jac", binary=TRUE)
head(spe.dj)
head(sqrt(spe.dj))
# Jaccard dissimilarity matrix using function dist()
spe.dj2 <- dist(spe, "binary")
head(spe.dj2)
# Jaccard dissimilarity matrix using function dist.binary()
spe.dj3 <- dist.binary(spe, method=1)
head(spe.dj3)
# Sørensen dissimilarity matrix using function dist.binary()
spe.ds <- dist.binary(spe, method=5)
head(spe.ds)
# Sørensen dissimilarity matrix using function vegdist()
spe.ds2 <- vegdist(spe, binary=TRUE)
head(spe.ds2) ; head(sqrt(spe.ds2))
# Ochiai dissimilarity matrix
spe.och <- dist.binary(spe, method=7)
```

```
head(spe.och)
  # The display of several values of the alternate versions of
  # the Jaccard and Sørensen distance matrices show differences.
  # Go back to the introduction of section 3.3 to understand the
  # differences.
```

Hint *Explore the arguments of the functions* **vegdist()**, **dist.binary()** *and* **dist()** *to see which coefficients are available. Some of them are available in more than one function. In* **dist()**, *argument binary produces (1 – Jaccard). In the help file of* **dist.binary()**, *be careful: the numbering of the coefficients does not follow Legendre and Legendre (1998), but Gower and Legendre (1986).*

Association matrices are generally intermediate entities, which are rarely examined directly. However, when there are not too many objects involved, it may be useful to display them in a way that emphasizes their main features. We suggest that you plot several dissimilarity matrices using the additional function **coldiss()**. A reordering feature is included in the function **coldiss()**, which uses the function **order.single()** of the **gclus** package to reorder each dissimilarity matrix so that similar sites are displayed close together along the diagonal. Therefore, you can compare the results obtained before and after reordering each matrix.

The package **gclus** is called within the **coldiss()** function so that it must have been installed prior to running the following code. Figure 3.1 shows an example.

```
# Graphical display of association matrices
# ****************************************

# The gclus package is required and may be called now, although
# it is called internally by coldiss()
library(gclus)

# Source the coldiss() function
source("coldiss.R")  # If necessary, add the path to the file

# Colour plots (also called heat maps, or trellis diagrams in
# the data analysis literature) using the coldiss() function
# ********************************************************

  # Colour plots of a dissimilarity matrix, without and with
  # ordering of distances. Screen output colours:
  # Magenta = dissimilarity close to 0 (maximum similarity)
  # Cyan = dissimilarity close to 1 (minimum similarity)
  # Usage:
  # coldiss(D=dist.object,nc=4,byrank=TRUE,diag=FALSE)
```

```
  # D should be a dissimilarity matrix
  # If D is a distance matrix with max(D) > 1, then D is divided
  # by max(D)
  # nc is the number of colours
  # byrank = TRUE           equal-sized categories
  # byrank = FALSE          equal-length intervals
  # If diag = TRUE, then site labels are displayed on the
  # diagonal

# Bray-Curtis dissimilarity matrix (on raw data)
# 4 colours with equal-length intervals (useful for comparisons)
coldiss(spe.db, byrank=FALSE, diag=TRUE)

# Same but on log-transformed data
coldiss(spe.dbln, byrank=FALSE, diag=TRUE)
```

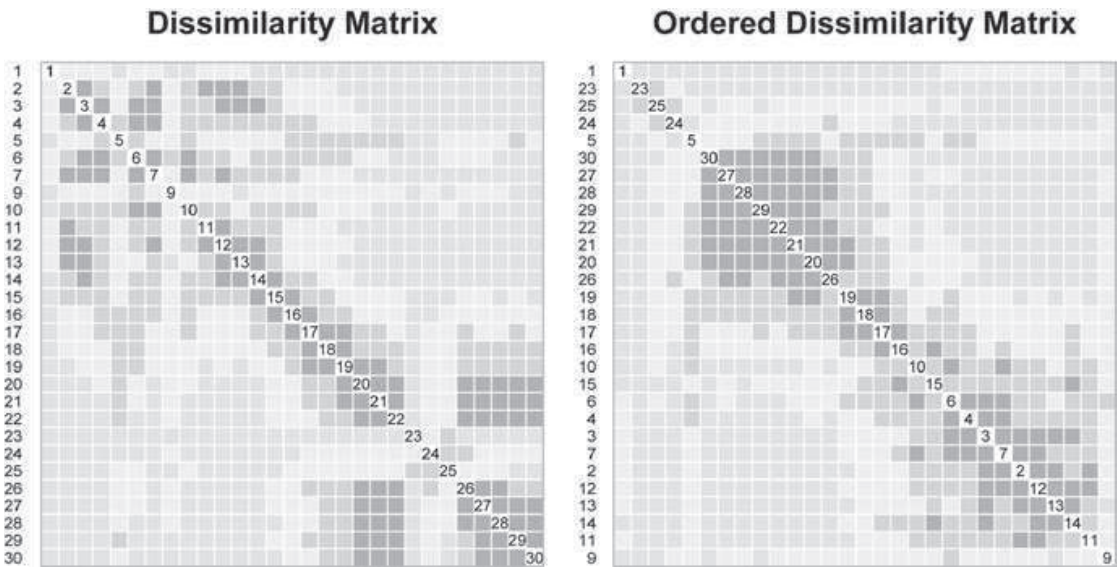

Fig. 3.1 Heat maps of a Bray–Curtis dissimilarity matrix computed on the raw fish data

Compare the two Bray–Curtis plots (raw and log-transformed data; the latter is not presented here). The differences are due to the log transformation. In the untransformed distance matrix, small differences in abundant species have the same importance as small differences in species with few individuals.

```
# Chord distance matrix
coldiss(spe.dc, byrank=FALSE, diag=TRUE)

# Hellinger distance matrix
coldiss(spe.dh, byrank=FALSE, diag=TRUE)
```

Compare the four colour plots. They all represent distance or dissimilarity matrices built upon quantitative abundance data. Are they similar?

```
# Jaccard distance matrix
coldiss(spe.dj, byrank=FALSE, diag=TRUE)

  # Compare the Jaccard plot with the previous ones. The Jaccard
  # plot is based on binary data. Does this influence the result?
  # Is the difference larger than between the various plots
  # based on quantitative coefficients?
```

Although these examples deal with species data, the Doubs transect is characterized by strong ecological gradients (e.g. oxygen, nitrate content; see Chap. 2). In such a well-defined context, it may be interesting to assume for discussion that species are absent for similar reasons from a given section of the stream, and compute an association matrix based on a symmetrical coefficient for comparison purposes. Here is an example using the simple matching coefficient S_1 (developed in Sect. 3.3.4).

```
# Simple matching dissimilarity
# (called the Sokal and Michener index in ade4)
spe.s1 <- dist.binary(spe, method=2)
coldiss(spe.s1^2, byrank=FALSE, diag=TRUE)

  # Compare this symmetrical dissimilarity matrix with the
  # Jaccard matrix. Which dissimilarities are the most affected
  # by taking, or not, double zeros into account?
```

The Code It Yourself corner #2

Write several lines *of code to compute Jaccard's "coefficient of community" (S_7) between sites 15 and 16 of the* `spe` *data frame. Sites 15 and 16 now have row numbers 14 and 15, respectively, because we deleted row 8 which is devoid of species.*

Jaccard's similarity index has the following formula for two objects x_1 *and* x_2:

$$S_{(x1,x2)} = a / (a + b + c)$$

where a is the number of double 1s, and b and c are the numbers of 0-1 and 1-0 combinations, respectively.

After the computation, convert the similarity value to a dissimilarity using the formulas used in **vegan** *and* **ade4** *(not the same formula).*

For the aficionados: display the code used in the **ade4** *function* **dist.binary()** *(just type* `dist.binary`*). Look how the quantities a, b, c and d are computed in just four lines for whole data frames. A fine example of programming elegance by Daniel Chessel and Stéphane Dray.*

3.3.3 Q Mode: Quantitative Data (Excluding Species Abundances)

For quantitative variables with a clear interpretation of double zeros, the queen of the symmetric distance measures is the *Euclidean distance* D_1. "*It is computed using Pythagora's formula, from site-points positioned in a* p*-dimensional space called a* metric *or* Euclidean space" (Legendre and Legendre 1998, p. 277).

The Euclidean distance has no upper limit, and its value is strongly influenced by the scale of each descriptor. For instance, changing the scale of a set of measurements from g/L to mg/L will multiply the Euclidean distance by 1000. Therefore, the use of the Euclidean distance on raw data is restricted to datasets that are dimensionally homogeneous, like geographic coordinates. Otherwise, D_1 is computed on standardized variables (z-scores). This also applies to situations, where one wishes to give the same weight to all variables in a set of dimensionally homogeneous descriptors.

Here, you could compute a matrix of Euclidean distances on the (standardized) environmental variables of our `env` dataset. We remove one variable, `das` (distance from the source), since it is a spatial rather than an environmental descriptor. The results are displayed using **`coldiss()`** (Fig. 3.2).

```
# Remove the 'das' variable from the env dataset
env2 <- env[,-1]

# Euclidean distance matrix of the standardized env2 data frame
env.de <- dist(scale(env2))
coldiss(env.de, diag=TRUE)
```

Hint *See how the environmental variables have been standardized "on the fly" using the function* **`scale()`**.

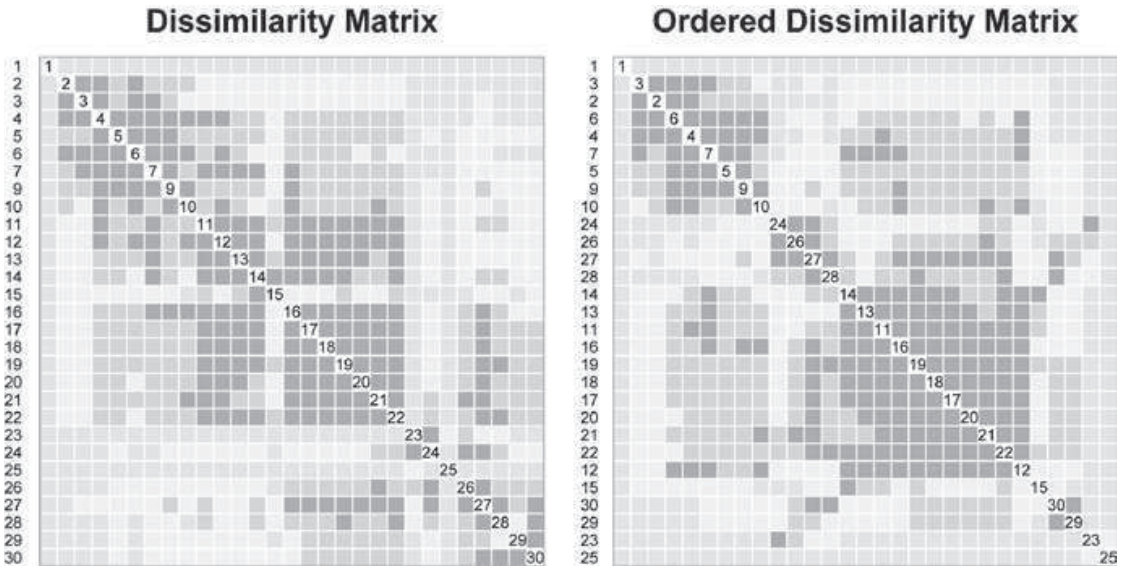

Fig. 3.2 Heat maps of a matrix of Euclidean distances on the standardized environmental variables

These plots of distance matrices can be used for a quick comparison. For instance, you could plot the Hellinger species-based distance matrix and the environmental distance matrix, both using equal-sized categories (`byrank=TRUE`, the default), in order to compare them visually:

```
# Hellinger distance matrix of the species data (equal-sized
# categories)
coldiss(spe.dh, diag=TRUE)

  # Compare the left-hand plots of this and the previous pair of
  # heat maps, since they present the sites in the same order.
  # Do you observe common features?
```

The Euclidean distance is of course appropriate to compute matrices of geographical distances based on variables giving the coordinates of the sites in any

units on a 1- or 2-dimensional orthogonal (Cartesian) system (such as cm, m, km or UTM coordinates). Coordinates in a spherical system (e.g. latitude–longitude) must be transformed prior to the computation of a Euclidean distance matrix. This transformation can be done by function **geoXY()** of the package **SoDA**. Note that *x*–*y* coordinates should *not* be standardized (only centred if necessary), since this would alter the ratio between the two dimensions.

In the following code lines, you will also compute a matrix of Euclidean distance on a single variable: `das`, the distance from the source of the river. This matrix thus represents the distances among sites along the river, while the matrix based on *x*–*y* coordinates represents the distance among points on a geographical map (as the crow flies, so to say).

```
# Euclidean distance matrix on spatial coordinates (2D)
spa.de <- dist(spa)
coldiss(spa.de, diag=TRUE)
# Euclidean distance matrix on distance from the source (1D)
das.df <- as.data.frame(env$das, row.names=rownames(env))
riv.de <- dist(das.df)
coldiss(riv.de, diag=TRUE)

 # Why are the x-y plot and the Euclidean distance from the
 # source plot so different?
```

3.3.4 Q Mode: Binary Data (Excluding Species Presence–Absence Data)

The simplest symmetrical similarity measure for binary data is the "simple matching coefficient" S_1. For each pair of sites, it is the ratio between the number of double 1s plus double 0s and the total number of variables.

The fish environment dataset is exclusively made of quantitative variables, so we create fictitious data to demonstrate the computation of S_1. We resort to this method from time to time, just to show how it is convenient to create datasets of known characteristics in **R**, for instance, for simulation purposes.

```
# Compute five binary variables with 30 objects each. Each
# variable has a predefined number of 0 and 1
# Variable 1: 10 x 1 and 20 x 0; the order is randomized
var1 <- sample(c(rep(1,10), rep(0,20)))
# Variable 2: 15 x 0 and 15 x 1, one block each
var2 <- c(rep(0,15), rep(1,15))
# Variable 3: alternation of 3 x 1 and 3 x 0 up to 30 objects
var3 <- rep(c(1,1,1,0,0,0),5)
```

```
# Variable 4: alternation of 5 x 1 and 10 x 0 up to 30 objects
var4 <- rep(c(rep(1,5), rep(0,10)),2)
# Variable 5: 16 objects with randomized distribution of 7 x 1
# and 9 x 0, followed by 4 x 0 and 10 x 1
var5.1 <- sample(c(rep(1,7), rep(0,9)))
var5.2 <- c(rep(0,4), rep(1,10))
var5 <- c(var5.1,var5.2)
# Variables 1 to 5 are put into a data frame
dat <- data.frame(var1,var2,var3,var4,var5)
dim(dat)

# Computation of a matrix of simple matching coefficients
# (called Sokal and Michener index in ade4)
dat.s1 <- dist.binary(dat, method=2)
coldiss(dat.s1, diag=TRUE)
```

3.3.5 *Q Mode: Mixed Types, Including Categorical (Qualitative Multiclass) Variables*

Among the association measures that can handle nominal data correctly, one is readily available in **R**: Gower's similarity S_{15}. This coefficient has been devised to handle data containing variables of various mathematical types, each variable receiving a treatment corresponding to its category. The final (dis)similarity between two objects is obtained by averaging the partial (dis)similarities computed for all variables separately. We use Gower's similarity as a symmetrical index; when a variable is declared as a *factor* in a data frame, the simple matching rule is applied, i.e. for each pair of objects the similarity is 1 for that variable if the factor has the same level in the two objects and 0 if the level is different. Gower's dissimilarity is computed using the function **`daisy()`** of package **`cluster`**. Avoid the use of **`vegdist()`** (method="`gower`"), which is appropriate for quantitative and presence–absence, but not for multiclass variables.

`daisy()` can handle data frames made of mixed-type variables, provided that each one is properly defined. Optionally, the user can provide an argument (in the form of a list) specifying the types of some or all variables in the dataset.

`gowdis()` of package **`FD`** is the most complete function to compute Gower's coefficient. It computes the distance for mixed variables, including asymmetrical binary variables. Variable weights can be specified. **`gowdis()`** implements Podani's (1999) extension to ordinal variables.

Let us again create an artificial dataset containing four variables: two random quantitative variables and two factors:

```
# Fictitious data for Gower (S15) index
# Random normal deviates with zero mean and unit standard
deviation
var.g1 <- rnorm(30,0,1)
# Random uniform deviates from 0 to 5
var.g2 <- runif(30,0,5)
# Factor with 3 levels (10 objects each)
var.g3 <- gl(3,10)
# Factor with 2 levels, orthogonal to var.g3
var.g4 <- gl(2,5,30)

 # Together, var.g3 and var.g4 represent a 2-way crossed
 # balanced design.

dat2 <- data.frame(var.g1,var.g2,var.g3,var.g4)
summary(dat2)
```

Hints *Function* **gl()** *is quite handy to generate factors, but unfortunately it uses numerals for the levels. Potential confusion would be avoided if alphanumerics were used instead of numbers.*

Note the use of **data.frame()** *to assemble the four variables. Unlike* **cbind()**, **data.frame()** *preserves the classes of the variables. Variables 3 and 4 thus retain their class* `"factor"`.

Let us first compute and view the complete S_{15} matrix. Then, repeat the computation using the two factors (`var.g3` and `var.g4`) only:

```
# Computation of a matrix of Gower dissimilarity using function
daisy()

# Complete data matrix (4 variables)
dat2.S15 <- daisy(dat2, "gower")
range(dat2.S15)
coldiss(dat2.S15, diag=TRUE)

# Data matrix with the two orthogonal factors only
dat2partial.S15 <- daisy(dat2[,3:4], "gower")
coldiss(dat2partial.S15, diag=TRUE)

# What are the dissimilarity values in the dat2partial.S15
# matrix?
levels(factor(dat2partial.S15))
```

```
  # The values correspond to pairs of objects that share the same
  # levels for 2, 1 or no factor. Pairs with the highest
  # dissimilarity values share no common levels.

# Computation of a matrix of Gower dissimilarity using gowdis()
# of package FD
library(FD) # If not already loaded
?gowdis
dat2.S15.2 <- gowdis(dat2)
range(dat2.S15.2)
coldiss(dat2.S15.2, diag=TRUE)

# Data matrix with the two orthogonal factors only
dat2partial.S15.2 <- gowdis(dat2[,3:4])
coldiss(dat2partial.S15.2, diag=TRUE)

# What are the dissimilarity values in the dat2partial.S15.2
# matrix?
levels(factor(dat2partial.S15.2))
```

3.4 R Mode: Computing Dependence Matrices Among Variables

Correlation-type coefficients must be used to compare variables in the R mode. These include the Pearson as well as the non-parametric correlation coefficients (Spearman, Kendall) for quantitative or semi-quantitative data, and contingency statistics for the comparison of qualitative variables (chi-square statistic and derived forms). For presence–absence data, binary coefficients such as the Jaccard, Sørensen and Ochiai coefficients can be used in the R mode to compare species.

3.4.1 R Mode: Species Abundance Data

Covariances as well as parametric and non-parametric correlation coefficients are often used to compare species distributions through space or time. Note that double-zeros as well as the joint variations in abundance contribute to increase the correlations. In the search for species associations, Legendre (2005) applied the transformations described in Sect. 3.5 in order to remove the effect of the total abundance per site prior to the calculation of parametric and non-parametric correlations. Some concepts of species association use only the positive covariances or correlations to recognize associations of co-varying species.

Besides correlations, the chi-square distance, which was used in the Q mode, can also be computed on transposed matrices (R mode) because it was originally developed to study contingency tables, which are transposable by definition.

The example below shows how to compute and display an R-mode chi-square dissimilarity matrix of the 27 fish species:

```
# R-mode dissimilarity matrix
# **************************

# Transpose matrix of species abundances
spe.t <- t(spe)

# Chi-square pre-transformation followed by Euclidean distance
spe.t.chi <- decostand(spe.t, "chi.square")
spe.t.D16 <- dist(spe.t.chi)
coldiss(spe.t.D16, diag=TRUE)

 # Can you identify groups of species on the right-hand display?
```

3.4.2 R Mode: Species Presence–Absence Data

For binary species data, the Jaccard (S_7), Sørensen (S_8) and Ochiai (S_{14}) coefficients can also be used in the R mode. Apply S_7 to the fish presence–absence data after transposition of the matrix (object `spe.t`):

```
# Jaccard index on fish presence-absence
spe.t.S7 <- vegdist(spe.t, "jaccard", binary=TRUE)
coldiss(spe.t.S7, diag=TRUE)

 # Compare the right-hand display with the one obtained with the
 # chi-square distance. Are the species groups consistent?
```

3.4.3 R Mode: Quantitative and Ordinal Data (Other than Species Abundances)

To compare dimensionally homogeneous quantitative variables, one can use either the covariance or Pearson's *r* correlation coefficient. Note, however, that these measures are *linear* so that they may perform poorly to detect monotonic but non-linear relationships among variables. If the variables are not dimensionally homogeneous, Pearson's *r* must be preferred to the covariance, since *r* is actually the covariance computed on standardized variables.

Comparison among ordinal variables, or among quantitative variables that may be monotonically but not linearly related, can be achieved using rank correlation coefficients like Spearman's ρ (rho) or Kendall's τ (tau).

Here are some examples based on the fish environmental data `env`. The function **`cor()`** (**`stats`** package) requires the *untransposed* matrix, i.e. the original matrix, where the variables are in columns. First example: Pearson's *r* (Fig. 3.3):

```
# Pearson r linear correlation among environmental variables
env.pearson <- cor(env)    # default method = "pearson"
round(env.pearson,2)

# Reorder the variables prior to plotting
env.o <- order.single(env.pearson)

# pairs: a function to plot a matrix of bivariate scatter
# diagrams and correlation coefficients. Correlations are given
# in the upper panel (with significance levels)
source("panelutils.R") # If necessary give path
op <- par(mfrow=c(1,1), pty="s")
pairs(env[,env.o], lower.panel=panel.smooth,
upper.panel=panel.cor, diag.panel=panel.hist, main="Pearson
 Correlation Matrix")
par(op)

 # Identify the variables correlated with variable 'das' (the
 # distance from the source). What story does that tell?
```

The same variables are now compared using Kendall's τ:

```
# Kendall tau rank correlation among environmental variables
env.ken <- cor(env, method="kendall")
env.o <- order.single(env.ken)
op <- par(mfrow=c(1,1), pty="s")
pairs(env[,env.o], lower.panel=panel.smooth,
upper.panel=panel.cor,method="kendall", diag.panel=panel.hist,
 main="Kendall Correlation Matrix")
par(op)

 # Judging by these plots of bivariate relationships,
 # would you favour the use of Kendall's tau or Pearson's r?
```

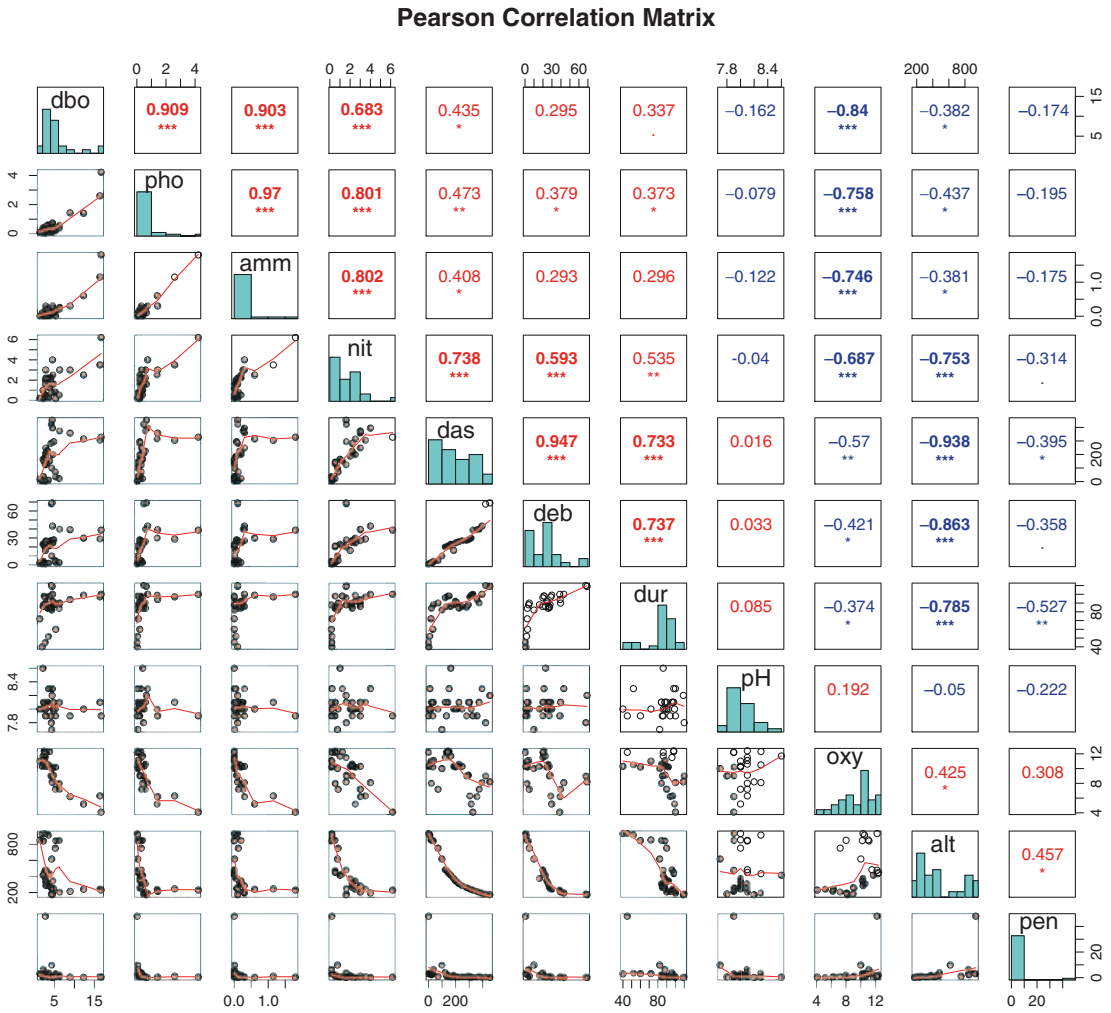

Fig. 3.3 Multipanel display of pairwise relationships between environmental variables with Pearson's *r* correlations

3.4.4 R Mode: Binary Data (Other than Species Abundance Data)

The simplest way of comparing pairs of binary variables is to compute a matrix of Pearson's *r*. In that case, Pearson's *r* is called the *point correlation coefficient* or Pearson's ϕ. That coefficient is closely related to the chi-square statistic for 2×2 tables without correction for continuity: $\chi^2 = n\phi^2$, where n is the number of objects.

3.5 Pre-transformations for Species Data

In Sect. 3.2.2, we explained why species abundance data should be treated in a special way, avoiding the use of double zeros as indications of resemblance among sites; linear methods, which explicitly or implicitly use the Euclidean distance among sites or the covariance or correlation among variables, are therefore not appropriate for such data. Unfortunately, many of the most powerful statistical tools available to ecologists, like ANOVA, *k*-means partitioning (see Chap. 4), principal component analysis (PCA, see Chap. 5) and redundancy analysis (RDA, see Chap. 6) *are* linear. Consequently, these methods were more or less "forbidden" to species data until Legendre and Gallagher (2001) showed than several asymmetrical association measures (i.e. measures that are appropriate to species data) can be obtained by two computational steps: a transformation of the raw data followed by the calculation of the Euclidean distance. These two steps preserve the asymmetrical distance among sites, therefore allowing the use of all linear methods of analysis with species data.

As is seen in the following chapters, in many cases one simply has to apply the pre-transformation to the species data, and then feed these to the linear methods of data analysis: PCA, RDA and so on.

Legendre and Gallagher proposed five pre-transformations of the species data. Four of them are available in **`vegan`** as arguments of the function **`decostand()`**: profiles of relative abundances by site (`"total"`), site normalization also called chord transformation (`"normalize"`), Hellinger transformation (`"hellinger"`) and chi-square double standardization (`"chi.square"`). See Sects. 2.2.4 and 3.3.1 for examples. All these transformations express the data as relative abundances per sites (site profiles) in some way; this removes from the data the total abundance per site, which is the response of the species to the total productivity of the sites. In the Hellinger transformation, the relative abundance values are square-rooted, which reduces more strongly the highest abundance values.

3.6 Conclusion

Although association matrices are in most cases intermediate entities, this chapter has shown that their computation deserves close attention. Many choices are available, and crucial decisions must be made at this step of the analytical procedure. The graphical tools presented in this chapter are precious helps to make these decisions, but great care must be taken to remain on solid theoretical ground. The success of the analytical steps following the computation of an association matrix depends on the choice of an appropriate association measure. Commonly-used distance functions in the Q mode, available in **R** packages, are presented in Table 3.1.

Table 3.1 Commonly-used distance and dissimilarity functions in Q mode available in R packages. The symbol ⇒ means that applying the function designed for quantitative data to presence–absence data produces the same result as computing the corresponding function designed for presence–absence data

Quantitative data		Presence–absence data
Community composition data		

Ruzicka dissimilarity
`vegdist(.,"jac")`
	⇒	Jaccard dissimilarity
		`vegdist(.,"jac", binary=TRUE)`
		`dist.binary(.,method=1)`

Hellinger distance `decostand(.,"hel")` followed by `vegdist(.,"euc")`	⇒	Ochiai dissimilarity `dist.binary(.,method=7)`
Chord distance `decostand(.,"norm")` followed by `vegdist(.,"euc")`	⇒	Ochiai dissimilarity `dist.binary(.,method=7)`
Bray–Curtis dissimilarity `vegdist(.,"bray")`	⇒	Sørensen dissimilarity `dist.binary(.,method=5)`
Chi-square distance `decostand(.,"chi.square")`		Chi-square distance (idem)
Canberra distance `vegdist(.,"canberra")`		
Other variables, mixed physical units		
Standardized variables: Euclidean distance `vegdist(.,"euc")`		Standardized variables: Simple matching coefficient `dist.binary(.,method=2)`
Non-standardized variables: Gower distance `daisy(.,"gower")`		

The similarity coefficients for presence–absence data and the Gower similarity should be transformed into distances using $\sqrt{1-S}$ to avoid the production of negative eigenvalues and imaginary eigenvectors in principal coordinate analysis. This transformation is made automatically by **ade4**, but not by **vegan**. In the table, functions **decostand()** and **vegdist()** belong to package **vegan**; function **dist.binary()** belongs to package **ade4**; function **daisy()** belongs to package cluster. Other **R** functions may be used for some distance coefficients. Some are mentioned in the course of the chapter. The Euclidean distance may be computed either by `vegdist(.,"euc")` as shown in the table, or by `dist(.)`.

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 a method 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 which is sometimes discrete (as in taxonomy), and most often perceived as continuous in ecology. This requires some degree of abstraction, but ecologists 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 other, 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 simultaneously the member of two genera: membership is binary (0 or 1). Some methods, less commonly

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_4, © Springer Science+Business Media, LLC 2011

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. Clustering is *not* a typical statistical method in that it does not test any hypothesis. 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.

One can recognize the following families of clustering methods (Legendre and Legendre 1998):

1. *Sequential or simultaneous algorithms.* Most clustering algorithms (i.e. effective methods for solving a problem using a finite sequence of instructions) 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 is to 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 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 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.

These categories are not represented equally in the ecologist's toolbox. Most methods presented below are sequential, agglomerative and hierarchical, but others, like *k*-means partitioning, are divisive and non-hierarchical. 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.

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.

We also discuss a recent method, multivariate regression tree (MRT) analysis, a technique of constrained divisive partitioning involving 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 divide the data of the first matrix.

Finally, a brief section is devoted to an example of fuzzy clustering, a non-hierarchical method 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 required  packages
library(ade4)
library (vegan)
library (gclus)
library(cluster)
library(RColorBrewer)
library(labdsv)
library(mvpart)
library(MVPARTwrap)  # Packages MVPARTwrap and rdaTest must be
                     # installed from zip files

# Import the data from CSV files
spe <- read.csv("DoubsSpe.csv", row.names=1)
env <- read.csv("DoubsEnv.csv", row.names=1)
spa <- read.csv("DoubsSpa.csv", row.names=1)
# Remove empty site 8
spe <- spe[-8,]
env <- env[-8,]
spa <- spa[-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 distances (or greatest similarities): the fusion of an object (or a group) with a group at a given similarity (or distance) 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 distance 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*; it is also called *minimum spanning tree*. This entity will be presented here and used in later analyses.

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:

```
# Species abundance data: compute matrix of chord distance among
# sites followed by single linkage agglomerative clustering
# ************************************************************
spe.norm <- decostand(spe, "normalize")
spe.ch <- vegdist(spe.norm, "euc")
spe.ch.single <- hclust(spe.ch, method="single")

# Plot a dendrogram using the default options
plot(spe.ch.single)

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

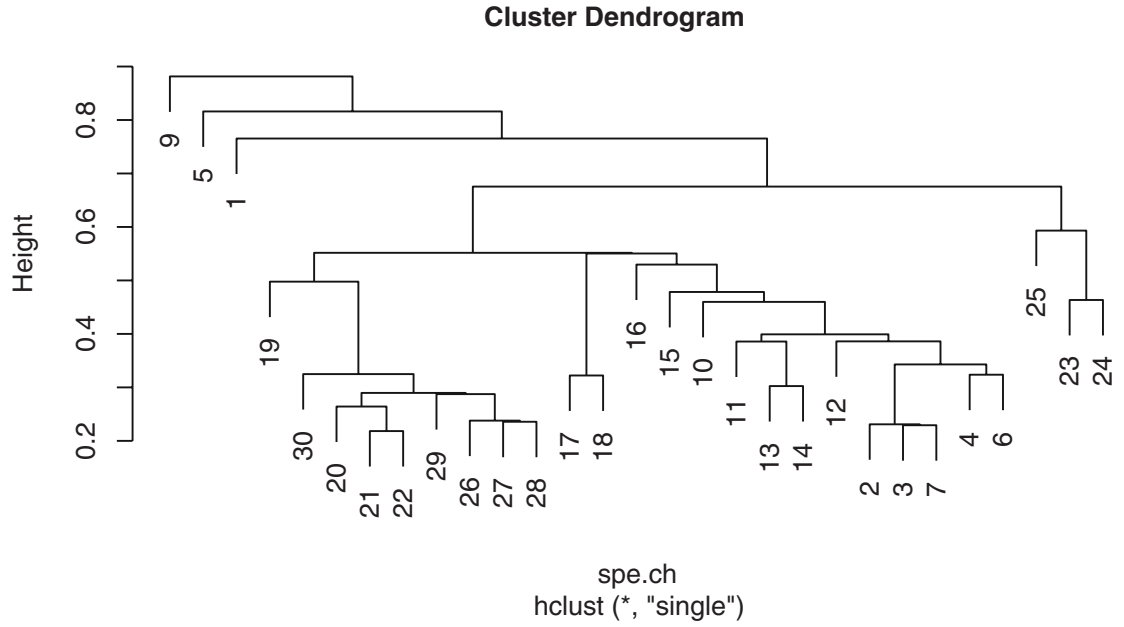

Fig. 4.1 Single linkage agglomerative clustering of a matrix of chord distance among sites (species data)

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 distance corresponding to that of the most distant pair of objects; thus, a fortiori, all members of both groups are linked (Fig. 4.2):

```
# Compute complete-linkage agglomerative clustering
# ************************************************
spe.ch.complete <- hclust(spe.ch, method="complete")
plot(spe.ch.complete)

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

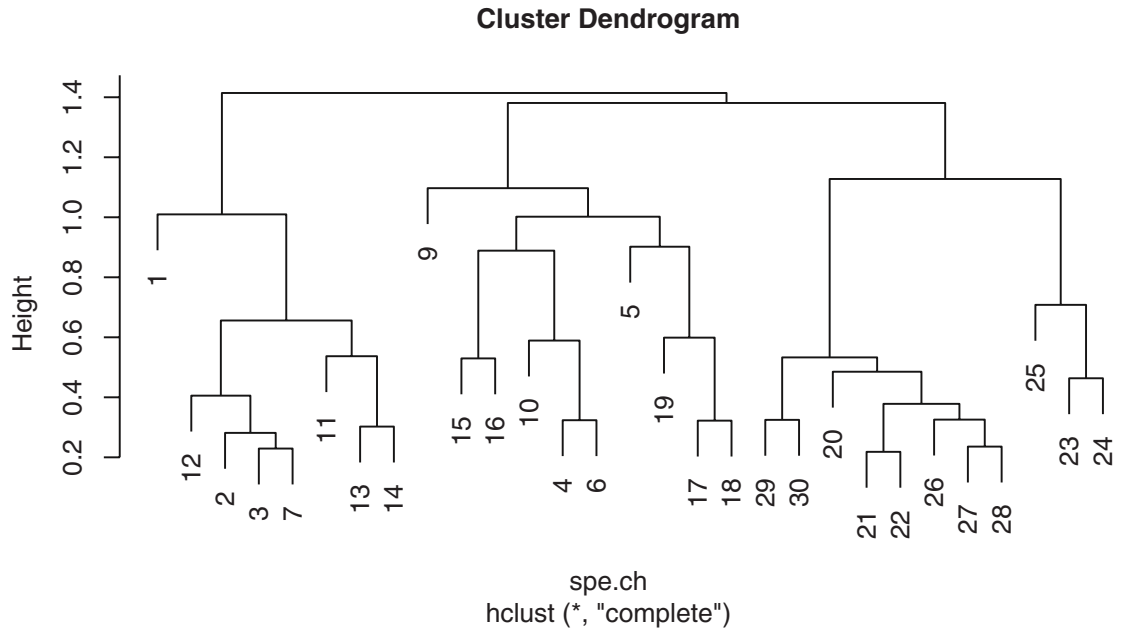

Fig. 4.2 Complete linkage agglomerative clustering of a matrix of chord distance among sites (species data)

The comparison between the two dendrograms 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 distance 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 distances. 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 distances between this object and all members of the group. When two groups join, they do it at the mean of the distances between all members of one group and all members of the other. Let us apply it to our data (Fig. 4.3):

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

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

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`"

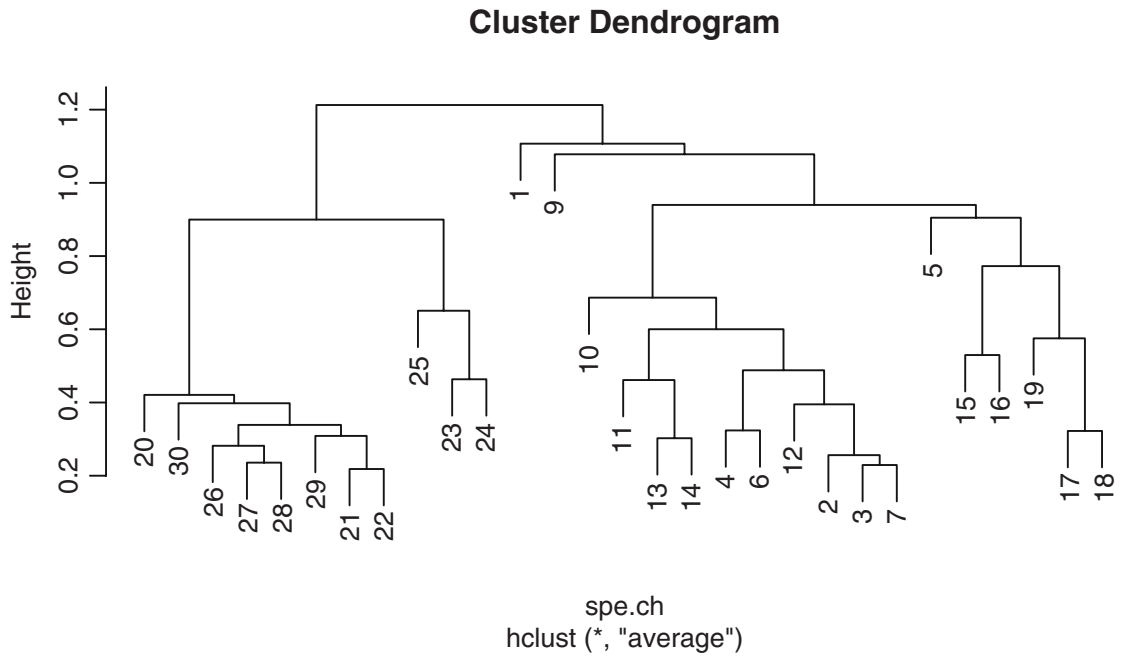

Fig. 4.3 UPGMA clustering of a matrix of chord distance among sites (species data)

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):

```
# Compute centroid clustering of the fish data
# ********************************************

spe.ch.centroid <- hclust(spe.ch, method="centroid")
plot(spe.ch.centroid)

  # The resulting dendrogram is an ecologist's nightmare.
  # Legendre and Legendre (1998, p. 341) explain how reversals
  # are produced and suggest to interpret them as polychotomies
  # rather than dichotomies.
```

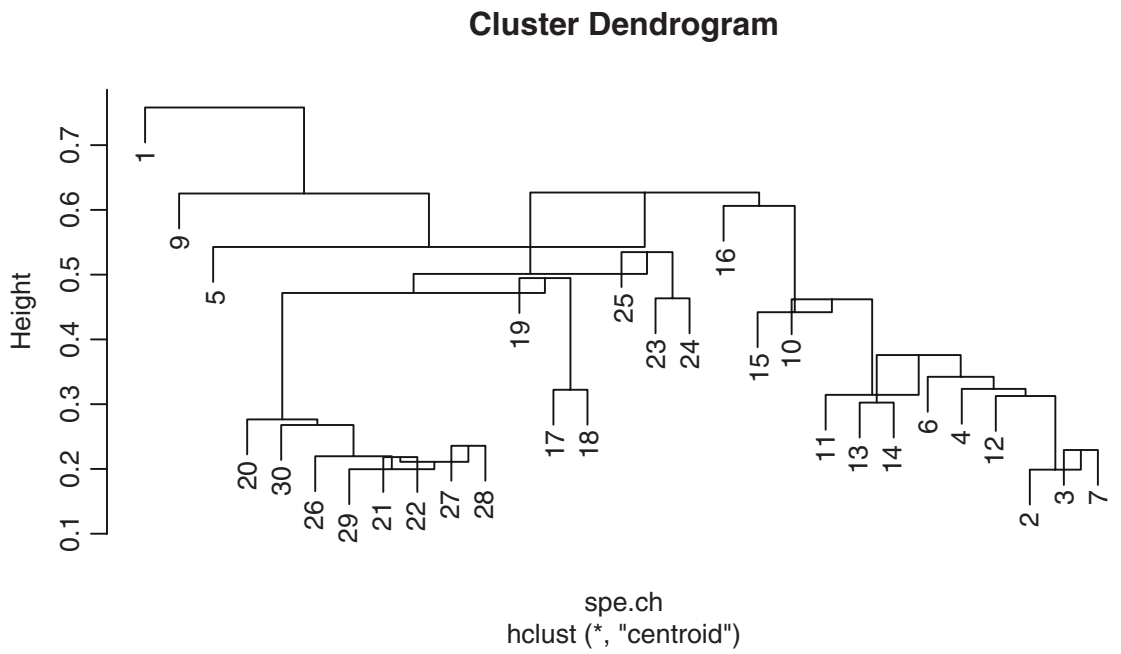

Fig. 4.4 UPGMC clustering of a matrix of chord distance among sites (species data)

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 cluster members divided by the number of objects. Note also that although the computation of within-group sums of squares (SS) is based on a Euclidean model, the Ward method produces meaningful results from distances that are Euclidean or not.

In **`hclust()`**, Ward's minimum variance clustering is obtained by using the argument `method="ward"` (Fig. 4.5):

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

```
# The distended look of the dendrogram is due to the use of
# squared distances as the ordinate scale by the plotting
# algorithm.
# To make the dendrogram more comparable to the others without
# affecting its topology), one should plot the square roots of
# the fusion levels instead (Fig. 4.5):

spe.ch.ward$height <- sqrt(spe.ch.ward$height)
plot(spe.ch.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.*

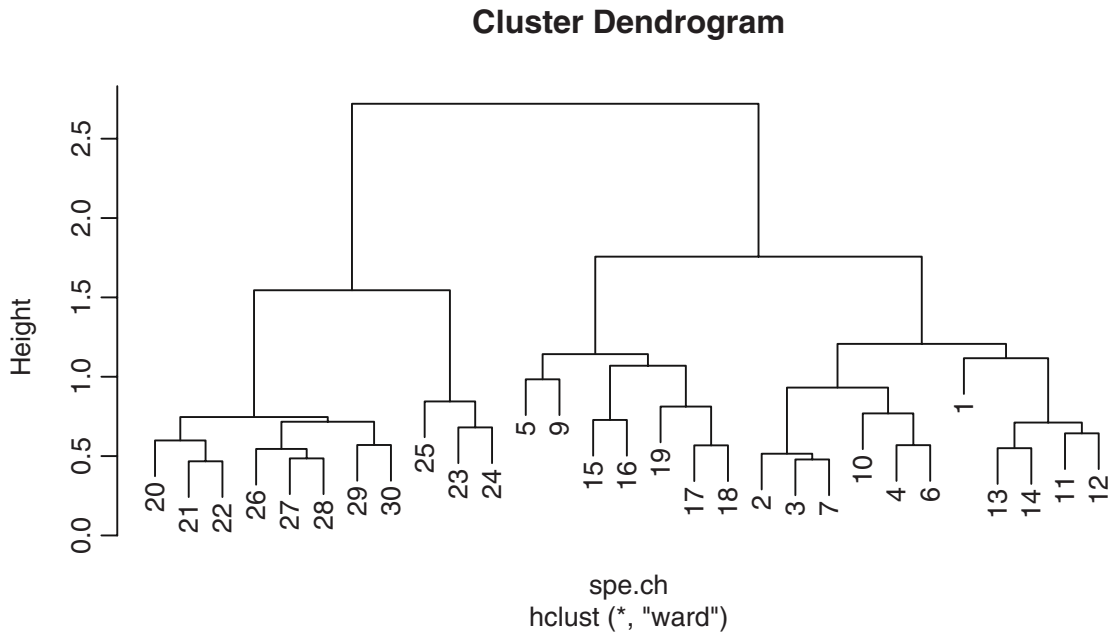

Fig. 4.5 Ward clustering of a matrix of chord distance among sites (species data)

4.6 Flexible Clustering

Lance and Williams (1966, 1967) proposed a model encompassing all the clustering methods seen above, which are obtained by changing the values of four parameters. See Legendre and Legendre (1998, p. 333). **`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`. See the help file of **`agnes()`** for more details.

4.7 Interpreting and Comparing Hierarchical Clustering Results

4.7.1 Introduction

Remember that 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(name_of_clustering_object)`.

This information can also be used to help interpret and compare clustering results. We 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 produced the best clustering model for the distance matrix.

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, independently of the data.

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

```
# Cophenetic correlation
# **********************

# 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 distance matrix and a set of cophenetic matrices obtained from various methods, one can draw Shepard-like diagrams (Legendre and Legendre 1998, p. 377) by plotting the original distances against the cophenetic distances (Fig. 4.6):

```
# 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 clustering", 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")
```

Another possible statistic for the comparison of clustering results is the Gower (1983) distance, computed as the sum of squared differences between the original 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 distance 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)
gow.dist.single
gow.dist.comp
gow.dist.UPGMA
gow.dist.ward
```

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, like a predetermined

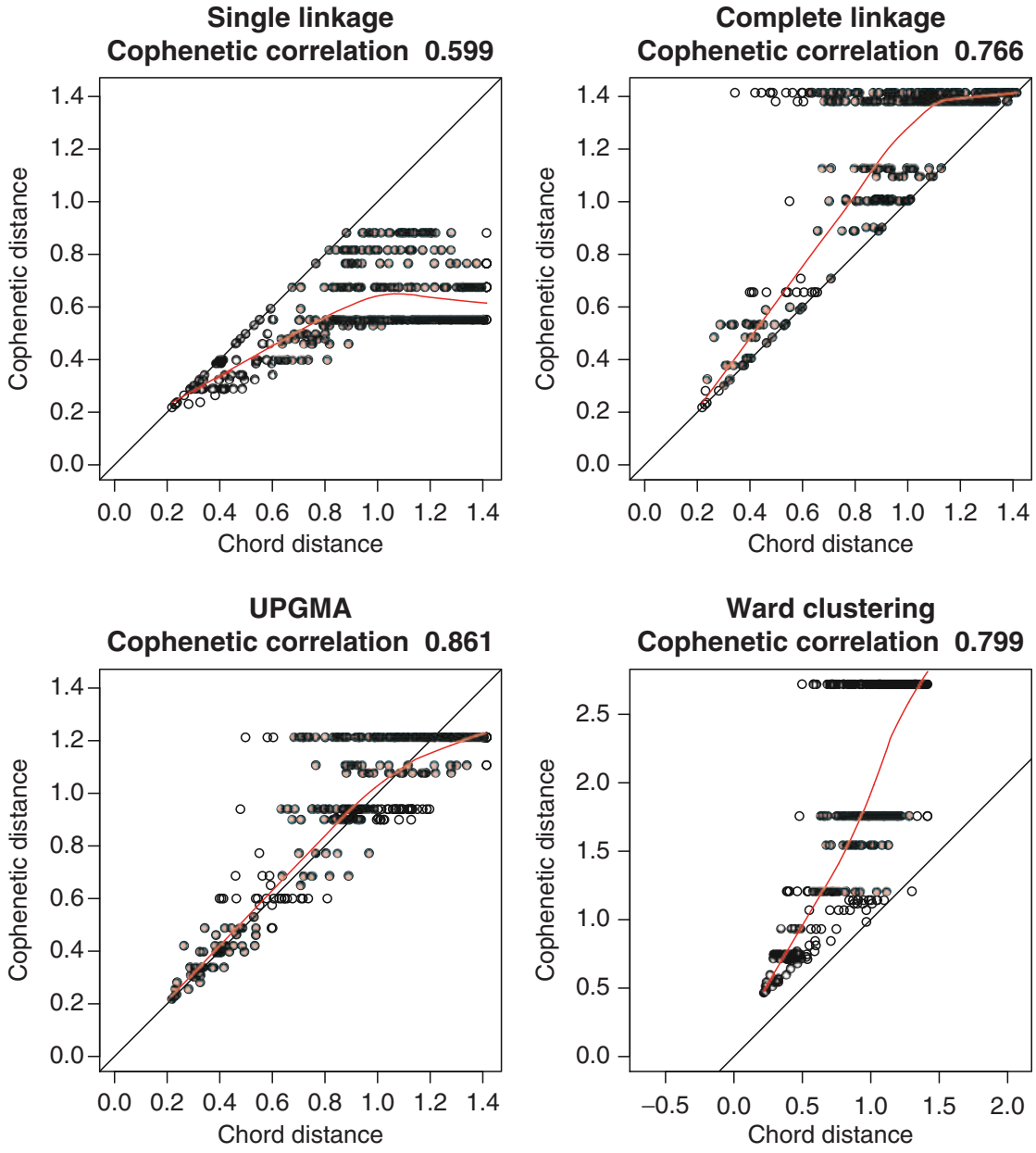

Fig. 4.6 Shepard-like diagrams comparing chord distances (species data) to four cophenetic distances. A LOWESS smoother shows the trend in each plot

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 help define cutting levels. Let us plot the fusion level values for some of the dendrograms produced above (Fig. 4.7).

```
# Graphs of fusion level values
# ****************************

par(mfrow=c(2,2))

# Plot the fusion level values of the single linkage clustering

summary(spe.ch.single)  # List of available results

plot(spe.ch.single$height, nrow(spe):2, type="S",
 main="Fusion levels - Chord - Single", ylab="k (number of
 clusters)", xlab="h (node height)", col="grey")
text(spe.ch.single$height, nrow(spe):2, nrow(spe):2, col="red",
 cex=0.8)
```

From right to left, this first graph shows clear jumps after each fusion between 2 and 6 groups. Go back to the dendrogram and cut it at the corresponding distances. Do the groups obtained always make sense? Do you obtain enough groups containing a substantial number of sites?

```
# 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)

 # Here the suggested numbers of groups may be different. Again,
 # try them on the dendrogram. Do these solutions make sense?

# 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)

 # Again the graphs look different from those above. Remember
 # that there is no single "truth" among these solutions. Each
 # one may provide some insight into the data.
```

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

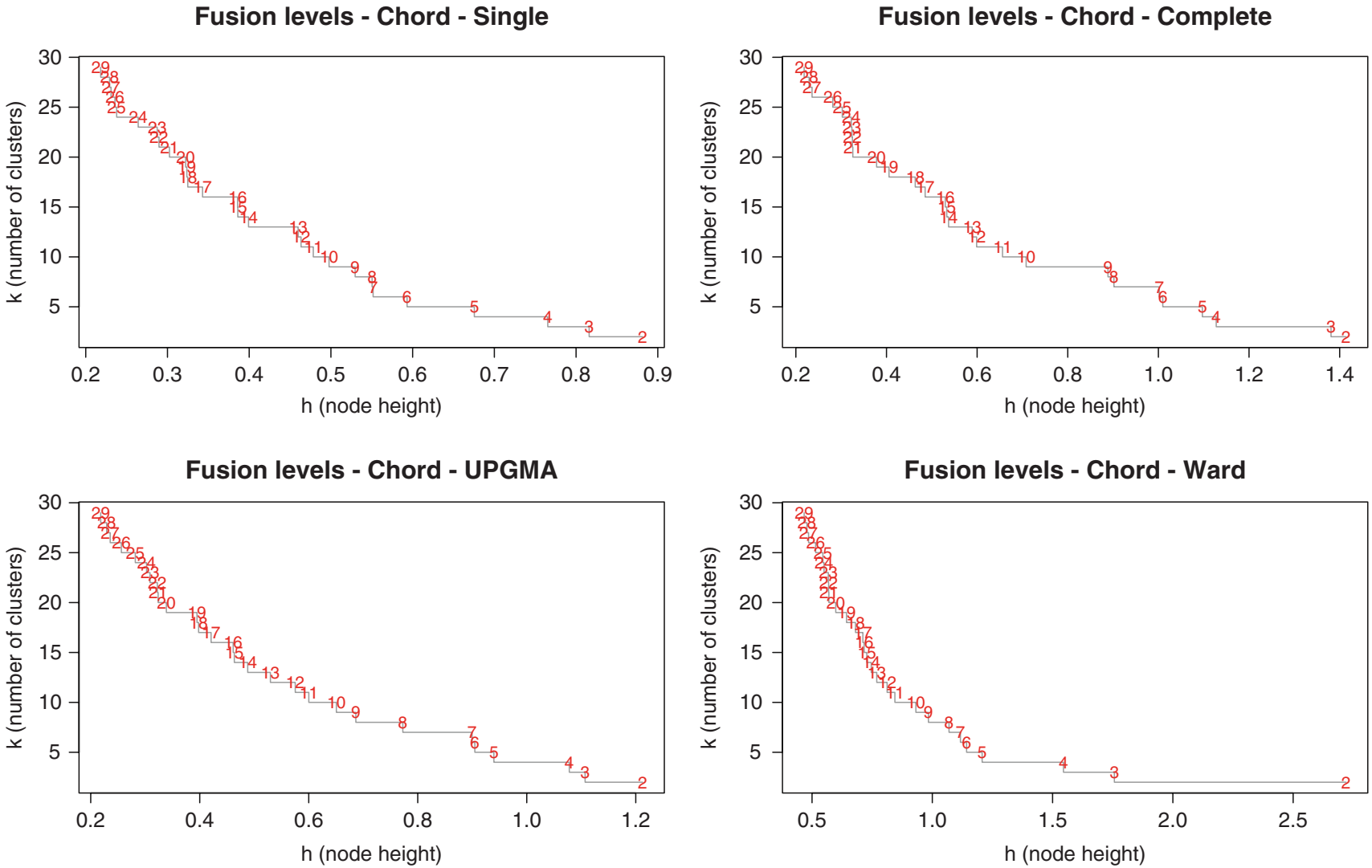

Fig. 4.7 Graphs of the fusion level values of four dendrograms

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:

```
# Cut the trees to obtain k groups and compare the group
# contents using 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
spebc.single.g <- cutree(spe.ch.single, k)
spebc.complete.g <- cutree(spe.ch.complete, k)
spebc.UPGMA.g <- cutree(spe.ch.UPGMA, k)
spebc.ward.g <- cutree(spe.ch.ward, k)

# Compare classifications by constructing contingency tables
# Single vs complete linkage
table(spebc.single.g, spebc.complete.g)
```

```
# Single linkage vs UPGMA
table(spebc.single.g, spebc.UPGMA.g)

# Single linkage vs Ward
table(spebc.single.g, spebc.ward.g)

# Complete linkage vs UPGMA
table(spebc.complete.g, spebc.UPGMA.g)

# Complete linkage vs Ward
table(spebc.complete.g, spebc.ward.g)

# UPGMA vs Ward
table(spebc.UPGMA.g, spebc.ward.g)

 # If two classifications had provided the same group contents,
 # one of the contingency tables would have shown only one non-
 # zero frequency value in each row and each column. This was
 # 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.
```

After the graphs of fusion levels, let us examine two other methods to help identify an appropriate number of groups: silhouette widths and Mantel comparison.

4.7.3.2 Graphs of Silhouette Widths

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

cluster to which is belongs, compared to the same measure computed for the next closest cluster (see Sect. 4.7.3.4). Silhouette widths range from −1 to 1 and can be averaged over all objects of a partition.

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

```
# Optimal number of clusters according to silhouette widths
# (Rousseeuw quality index)
# ********************************************************

# Plot average silhouette widths (using Ward clustering) for all
# partitions except for the trivial partition in a single group
# (k=1).

# First, create an empty vector in which the asw values
# will be written
asw <- numeric(nrow(spe))

# Second, retrieve and write the asw values into the vector
for (k in 2:(nrow(spe)-1)) {
 sil <- silhouette(cutree(spe.ch.ward, k=k), spe.ch)
 asw[k] <- summary(sil)$avg.width
 }

# Best (largest) silhouette width
k.best <- which.max(asw)

# The plot is produced by function plot.silhouette {cluster}
plot(1:nrow(spe), asw, type="h", main="Silhouette-optimal number
 of clusters, Ward", xlab="k (number of groups)", 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)

cat("", "Silhouette-optimal number of clusters k =", k.best,
"\n", "with an average silhouette width of", max(asw), "\n")

 # The message printed is:

# Silhouette-optimal number of clusters k = 2
# with an average silhouette width of 0.3658319

 # As it often happens, this criterion has selected 2 groups as
 # the optimal number. However, a partition into 4 groups seems
 # more interesting in ecological terms.
```

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

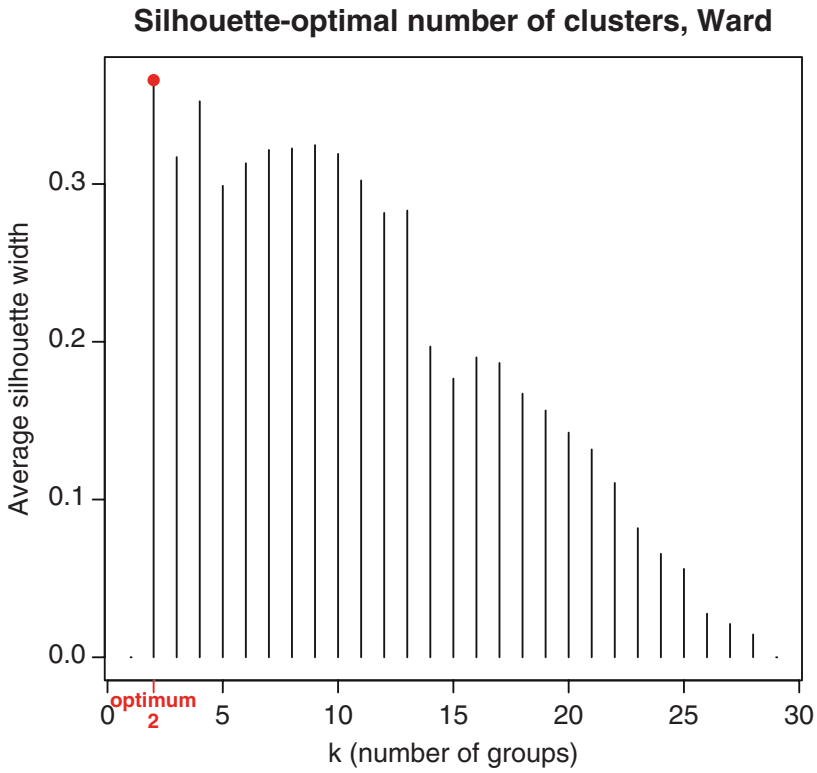

Fig. 4.8 Bar plot showing the average silhouette widths for $k = 2$–29 groups. The best partition by this criterion is the one with the largest average silhouette width

At each fusion level, 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 highest, that is, the largest average silhouette width. A bar plot is drawn (Fig. 4.8).

4.7.3.3 Comparison Between the Distance Matrix and Binary Matrices Representing Partitions

This technique compares the original distance matrix to binary matrices computed from the dendrogram cut at various levels (and representing group allocations); it then chooses the level where the matrix (Mantel) correlation between the two is the highest. Here the Mantel correlation is meant in its simplest sense, i.e. the equivalent of a Pearson r correlation between the values in the distance matrices. Tests are impossible here since the matrices are not independent of one another (the matrices corresponding to the partitions are derived from the same original distance matrix).

To compute the binary dissimilarity matrices representing group membership, we use a small function to compute a binary distance matrix from a vector-defining group. The results are shown in Fig. 4.9.

```
# Optimal number of clusters according to Mantel statistic
# (Pearson)
# ******************************************************

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

# Run based on the Ward clustering

kt <- data.frame(k=1:nrow(spe), r=0)

for (i in 2:(nrow(spe)-1)) {
 gr <- cutree(spe.ch.ward, i)
 distgr <- grpdist(gr)
 mt <- cor(spe.ch, distgr, method="pearson")
 kt[i,2] <- mt
}

kt
k.best <- which.max(kt$r)

# The plot is produced by function plot.silhouette {cluster}
plot(kt$k, kt$r, type="h", main="Mantel-optimal number of
 clusters - Ward", xlab="k (number of groups)", 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)
```

Hint *Since the Mantel correlation coefficient is algebraically equivalent to the Pearson correlation coefficient and no tests can be done here, the matrix correlation can be computed using the function* **`cor()`**. *Mantel functions are also available in the* **`vegan`**, **`ade4`** *and* **`ape`** *packages.*

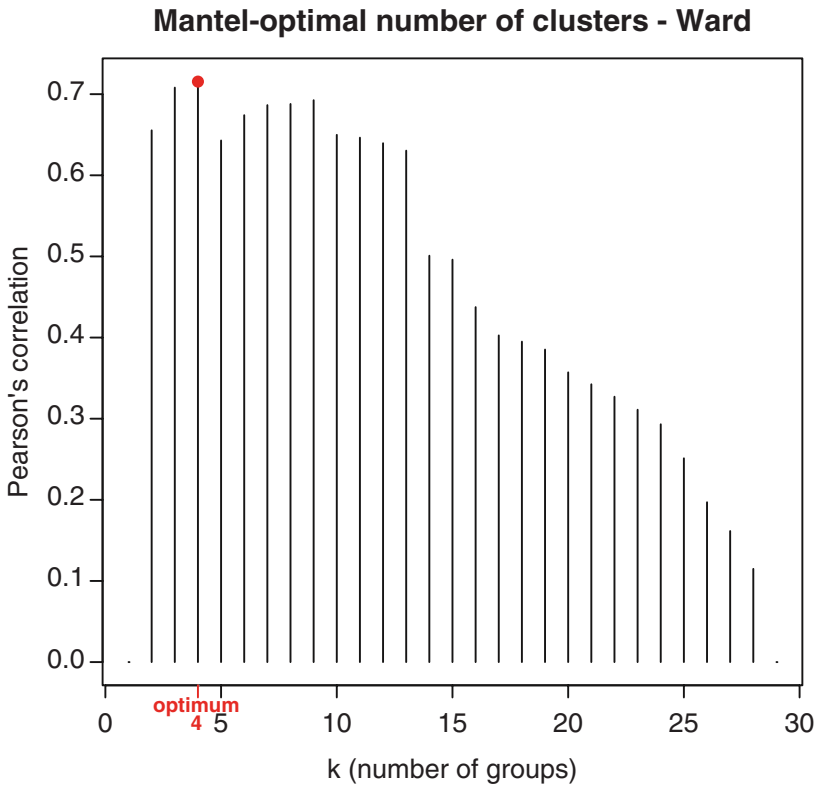

Fig. 4.9 Bar plot showing the correlations between the original distance matrix and binary matrices computed from the dendrogram cut at various levels

4.7.3.4 Silhouette Plot of the Final Partition

In our example, a number of groups that seems to be a good compromise between too few and too many is $k=4$. 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 reasonably 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. 4.10).

```
# Silhouette plot of the final partition
# *************************************
# Choose the number of clusters
k <- 4
```

```
# Silhouette plot
cutg <- cutree(spe.ch.ward, k=k)
sil <- silhouette(cutg, spe.ch)
silo <- sortSilhouette(sil)
rownames(silo) <- row.names(spe)[attr(silo,"iOrd")]
plot(silo, main="Silhouette plot - Chord - Ward",
  cex.names=0.8, col=cutg+1, nmax.lab=100)

  # Groups 1 and 3 are the most coherent, while group 2
  # contains misclassified objects.
```

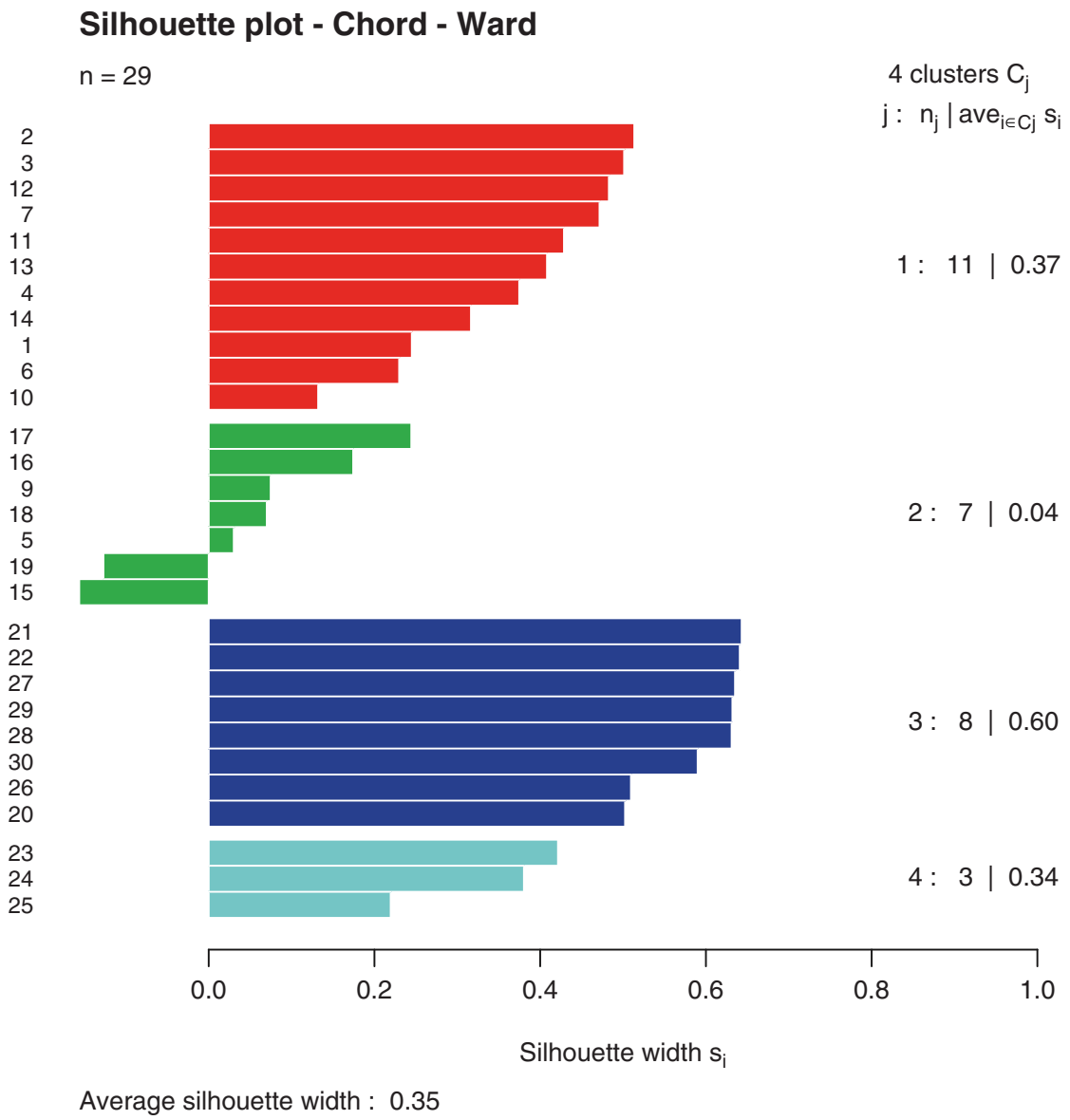

Fig. 4.10 Silhouette plot of the final, four-group partition from the Ward clustering

4.7.3.5 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. 4.11). Try the code, compare the results with it, and use the help files to understand the arguments used. Note also the use of a homemade function, **`hcoplot()`**.

```
# Final dendrogram with the selected groups
# ****************************************

# Reorder dendrogram from hclust(). reorder.hclust() reorders
# objects so that their order in the dissimilarity matrix is
# respected as much as possible. This does not affect the
# topology of the dendrogram.

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:

source("hcoplot.R") # hcoplot.R must be in the working directory
hcoplot(spe.ch.ward, spe.ch, k=4)
```

Hints *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.*

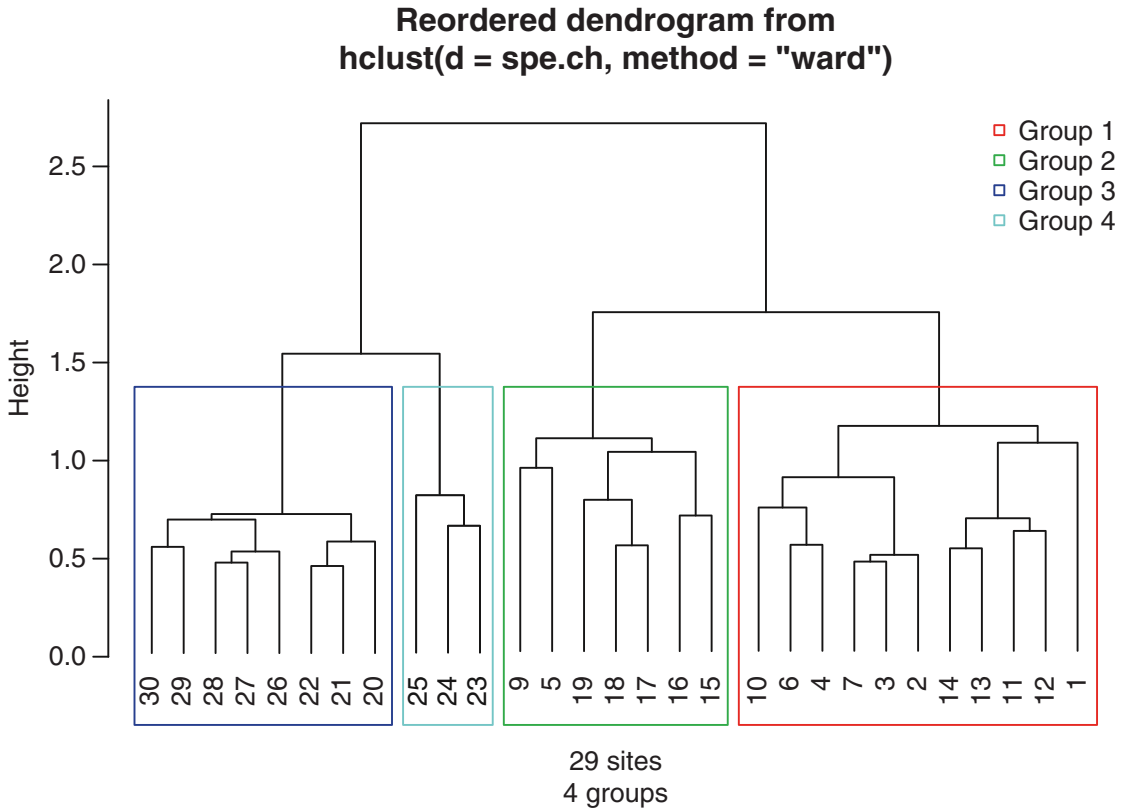

Fig. 4.11 Final dendrogram with *boxes* around the four selected groups. Species data

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

4.7.3.6 Spatial Plot of the Clustering Result

The code below allows one to plot the clusters on a map representing the river. This is a very useful way of representing results for spatially explicit data (Fig. 4.12).

```
# Plot of the 4 Ward clusters on a map of the Doubs river
# ******************************************************

# Map of the Doubs river (see Chapter 2)
plot(spa, asp=1, type="n", main="Four Ward groups",
  xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")
text(50, 10, "Upstream", cex=1.2, col="red")
```

```
text(25, 115, "Downstream", cex=1.2, col="red")
# Add the four groups
grw <- spebc.ward.g
k <- length(levels(factor(grw)))
for (i in 1:k) {
 points(spa[grw==i,1], spa[grw==i,2], pch=i+20, cex=3,
 col=i+1, bg=i+1)
   }
text(spa, row.names(spa), cex=0.8, col="white", font=2)
legend("bottomright", paste("Group",1:k), pch=(1:k)+20,
 col=2:(k+1), pt.bg=2:(k+1), pt.cex=2, bty="n")

  # Go back to the maps of four fish species (Chapter 2).
  # Compare these to the map just created above.
```

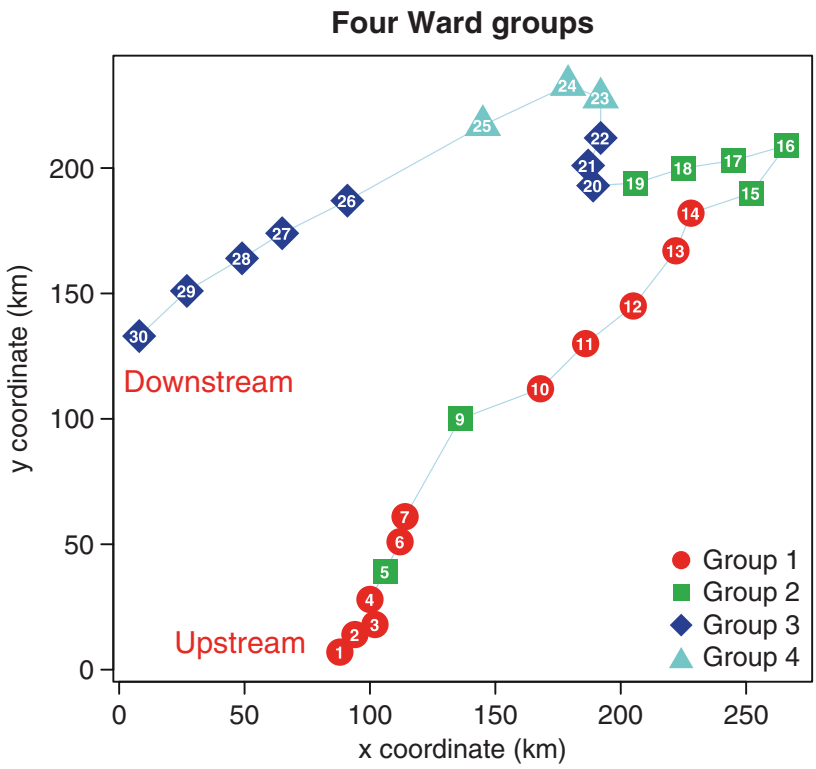

Fig. 4.12 The four Ward clusters on a map of the Doubs river

4.7.3.7 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. 4.13):

```
# Heat map
# ********

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

  # Observe how, thanks to the ordering, most "hot" (dark, or
  # red on the computer output) values representing high
  # similarities are located close to the diagonal (the diagonal
  # itself is trivial).
```

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 of the* **`as.dendrogram()`** *function, where examples are found.*

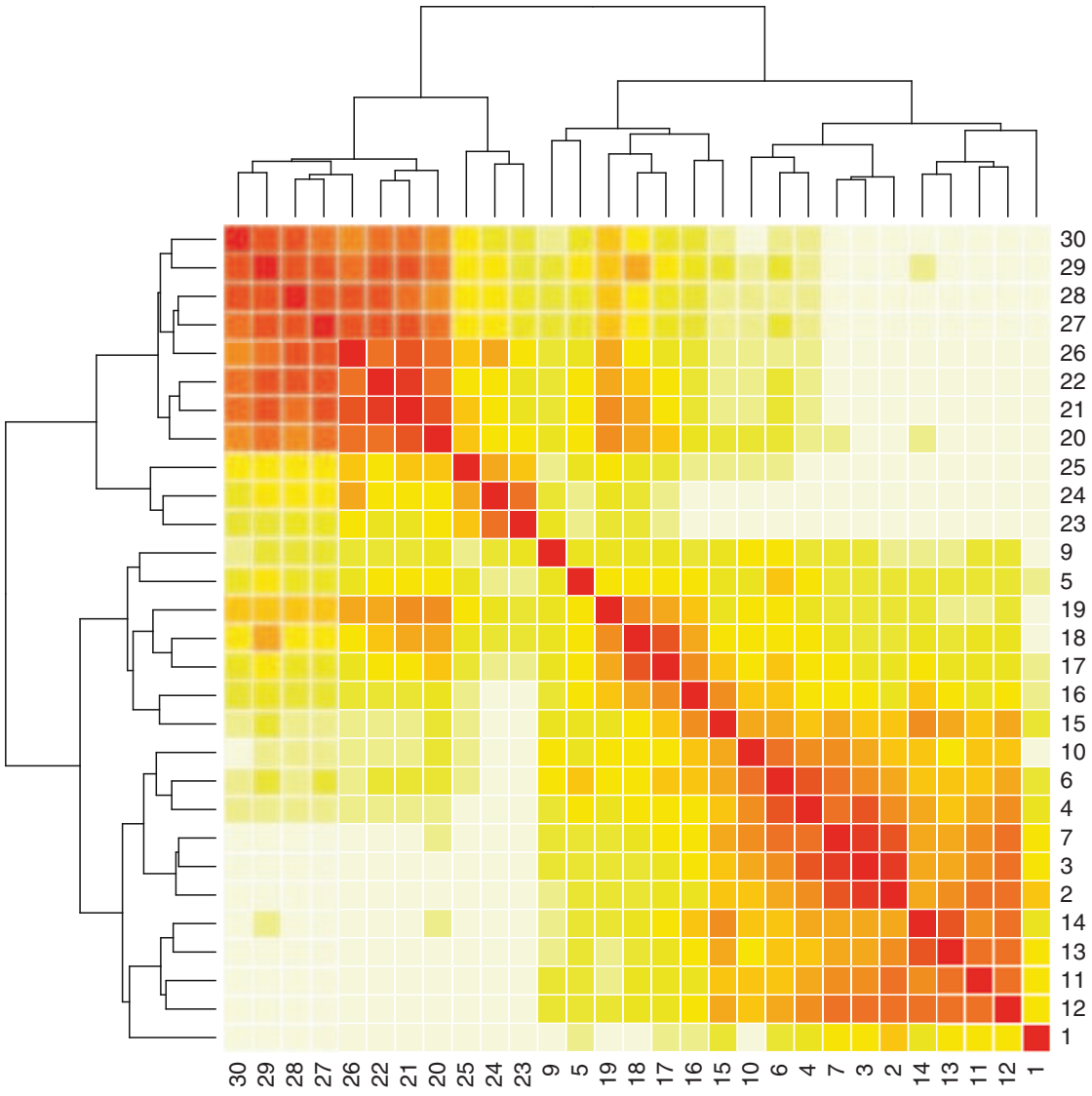

Fig. 4.13 Heat map of the distance matrix reordered according to the dendrogram

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. This way of ordering species can also be used in a heat map with colour intensities proportional to the species abundances (Fig. 4.14):

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

# 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")
```

4.8 Non-hierarchical Clustering

Partitioning is 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 it is often random. In that case, the analysis is run a large number of times with different initial configurations to find the best solution.

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

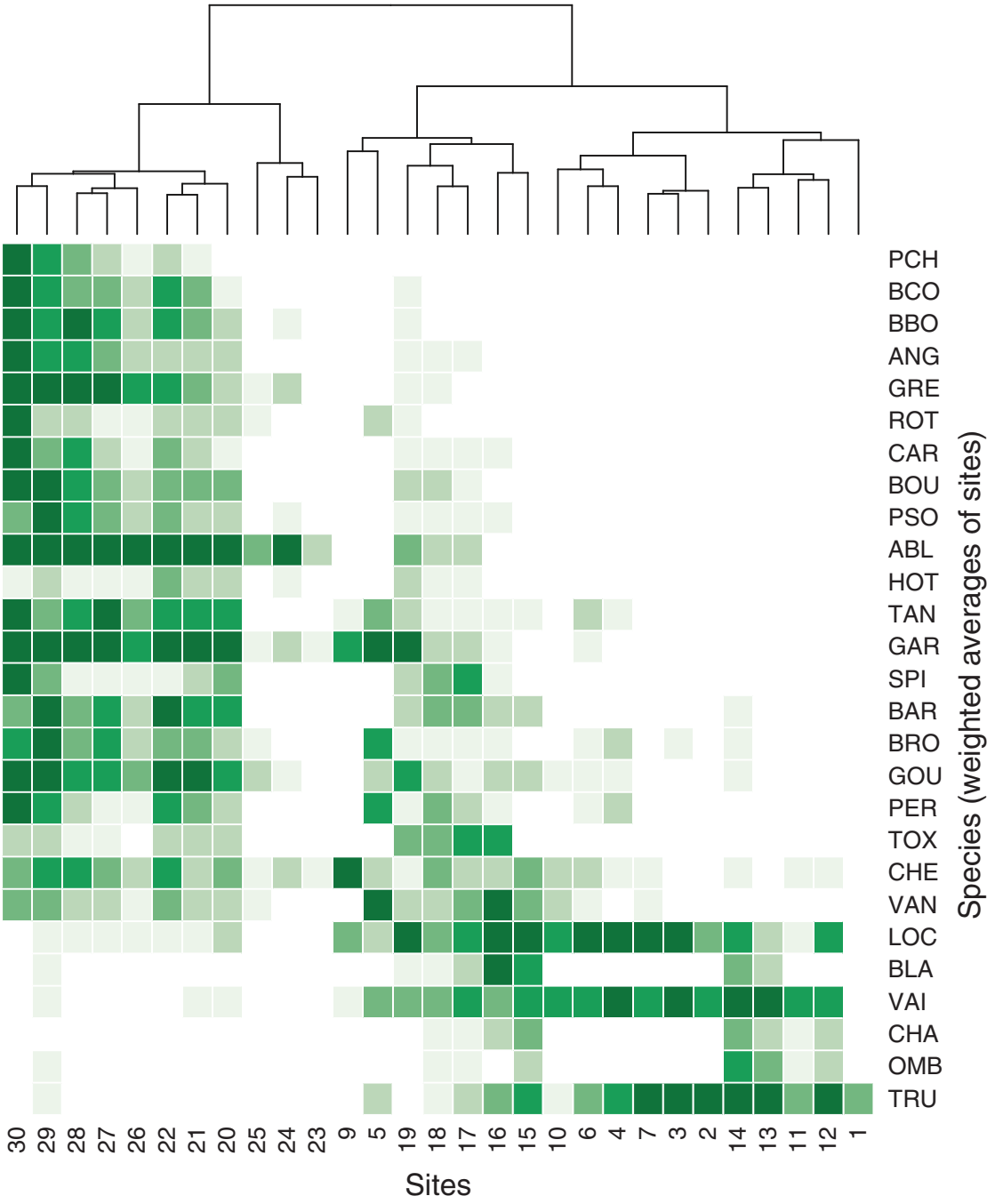

Fig. 4.14 Heat map of the doubly ordered community table, with dendrogram

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-groups sums-of-squares. This quantity is the sum, over the k groups, of the sums of the squared

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.

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). A solution is to pre-transform the species data. To remain coherent with the previous sections, we can use our "normalized" species data. 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, let us 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
# ********************************************************

spe.kmeans <- kmeans(spe.norm, centers=4, nstart=100)

  # Note: running the function again may produce slightly
  # different results as each of the 'nstart' runs starts with a
  # different random configuration

# Comparison with the 4-group classification derived from Ward
clustering:
table(spe.kmeans$cluster, spebc.ward.g)

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

To compute *k*-means partitioning from distance indices that cannot be obtained by a transformation of the raw data followed by calculation of the Euclidean distance, for example the Bray–Curtis dissimilarity, one has to compute first a rectangular data table with *n* rows by principal coordinate analysis (PCoA, Chap. 5) of the distance matrix, then use that rectangular table as input to *k*-means partitioning. For the Bray–Curtis dissimilarity, one has to compute the PCoA on the square root of the Bray–Curtis 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 is the best solution 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 help 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 data set, asking for two to ten groups and the simple structure index criterion for clustering quality, followed by a plot of the results (Fig. 4.15).

```
# k-means partitioning, 2 to 10 groups
# ************************************

spe.KM.cascade <- cascadeKM(spe.norm, inf.gr=2, sup.gr=10,
 iter=100, criterion="ssi")
plot(spe.KM.cascade, sortg=TRUE)

 # The plot shows the group attributed to each object for each
 # partition (rows of the graph). 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. How many groups does this cascade propose as the "best"
 # solution? If one prefers a larger number of groups, what
 # would be the next best solution?
```

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.*

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.

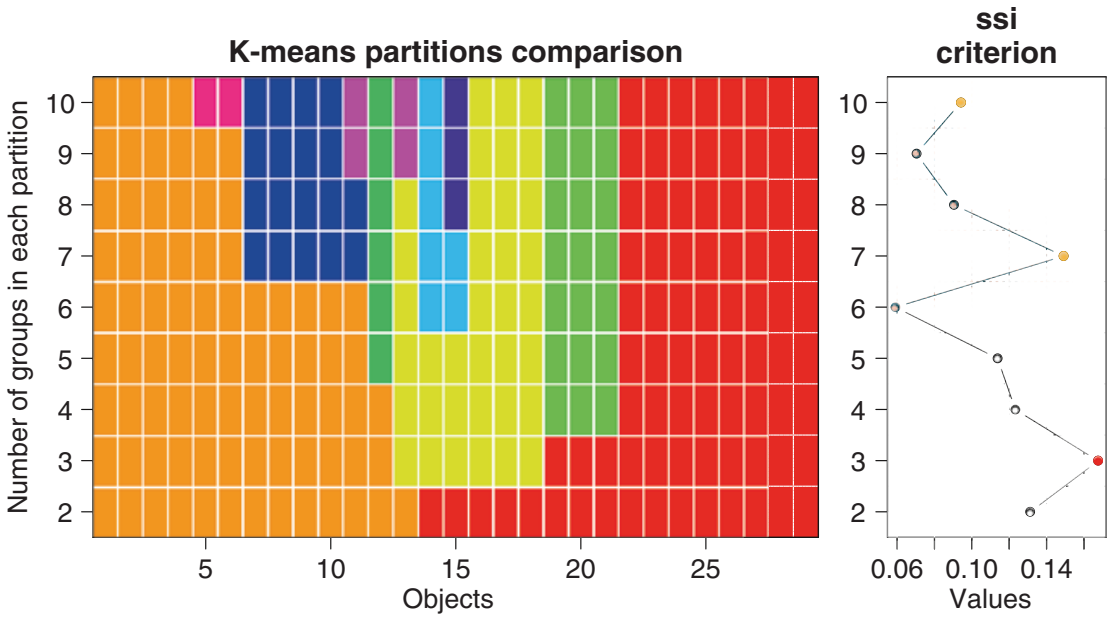

Fig. 4.15 *k*-means cascade plot showing the group attributed to each object for each partition

```
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 ssi are good
  # criteria to find the optimal value of k.

spe.KM.cascade$partition

  # 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. Here is an example based upon the *k*-means four-group partition.

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

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

4.8.2 Partitioning Around Medoids

Partitioning around medoids (Chapter 2 in Kaufman and Rousseeuw 1990) *"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** help 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. As implemented in ***R***, **pam()** (package **cluster**) accepts raw data or dissimilarity matrices (an advantage over **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. 4.16).

```
# Partitioning around medoids (PAM)
# Computed on the chord distance matrix
# ************************************

# Choice of the number of clusters
asw <- numeric(nrow(spe))

# Loop to compute average silhouette width for 2 to 28 groups.
# asw means "average silhouette width"
for (k in 2:(nrow(spe)-1))
 asw[k] <- pam(spe.ch, k, diss=TRUE)$silinfo$avg.width
k.best <- which.max(asw)
cat("", "Silhouette-optimal number of clusters k =", k.best,
 "\n", "with an average silhouette width of", max(asw), "\n")
plot(1:nrow(spe), asw, type="h", main="Choice of the number of
 clusters", xlab="k (number of groups)", 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 groups
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, spebc.ward.g)
table(spe.ch.pam.g, spe.kmeans$cluster)
```

```
  # 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))
plot(silhouette(spe.kmeans$cluster,spe.ch), main="Silhouette
 plot - k-means", cex.names=0.8, col= spe.kmeans$cluster+1)
plot(silhouette(spe.ch.pam), main="Silhouette plot - PAM",
 cex.names=0.8, col=spe.ch.pam$silinfo$widths+1)

  # On the basis on this plot, you should be able to tell which
  # solution (PAM or k-means) has a better silhouette.
  # You could also compare this result with the silhouette
  # profile of the Ward clustering produced earlier.
```

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).

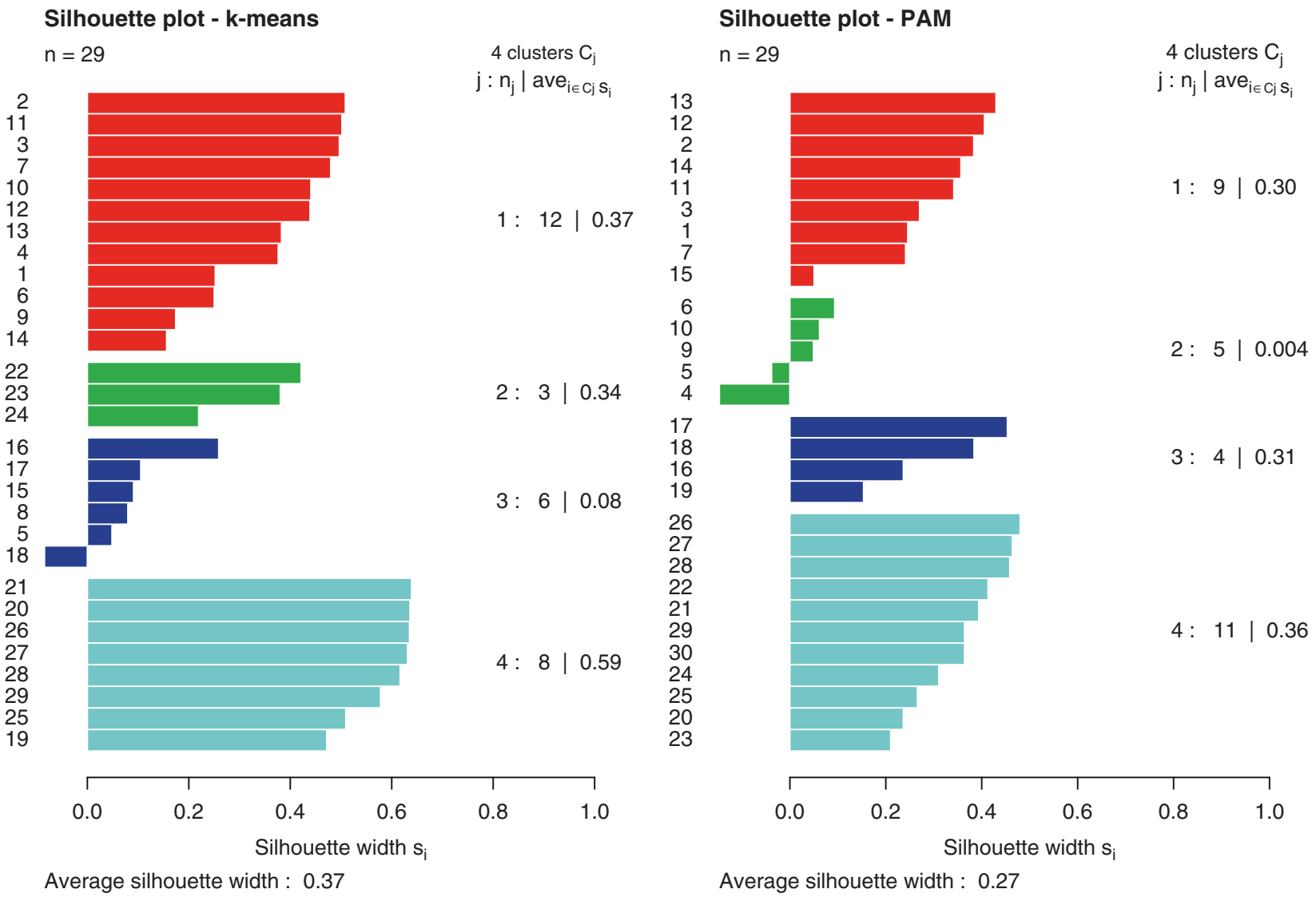

Fig. 4.16 Silhouette plots of the *k*-means and PAM results

As a final bonus, you could create a map of the four *k*-means clusters similar to that created for the Ward clustering. The code is the same, except for the fact that the object containing the clustering information is now `spe.kmeans$cluster` for results obtained from **`kmeans()`**, or in object `spe.KM.cascade$partition` for calculations done with **`cascadeKM()`**:

```
# Plot of the 4 k-means clusters on a map of the Doubs river
# ***********************************************************

plot(spa, asp=1, type="n", main="Four k-means groups",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")
text(50, 10, "Upstream", cex=1.2, col="red")
text(25, 115, "Downstream", cex=1.2, col="red")

grKM <- spe.kmeans$cluster
k <- length(levels(factor(grKM)))

for (i in 1:k) {
 points(spa[grKM==i,1], spa[grKM==i,2], pch=i+20, cex=3,
 col=i+1, bg=i+1)
   }

text(spa, row.names(spa), cex=0.8, col="white", font=2)
legend("bottomright", paste("Group",1:k), pch=(1:k)+20,
 col=2:(k+1), pt.bg=2:(k+1), pt.cex=2, bty="n")
```

4.9 Comparison with Environmental Data

All the methods above have been presented with examples on species abundance data. They can 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 a reversed point of view, the clusters obtained from the species 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 variable separately, followed either by a classical ANOVA or by a non-parametric Kruskal–Wallis test. Boxplots of the environmental variables (after some simple transformations to improve normality) for the four *k*-means groups are also provided (Fig. 4.17).

```
# Relationships between fish clusters and 4 environmental
# variables based on the k-means clustering results (4 groups)
# ************************************************************

attach(env)

# Boxplots of quantitative environmental variables:
# Altitude, Slope, Oxygen, and Ammonium

par(mfrow=c(2,2))

boxplot(sqrt(alt) ~ spe.kmeans$cluster, main="Altitude", las=1,
  ylab="sqrt(alt)", col=2:5, varwidth=TRUE)
boxplot(log(pen) ~ spe.kmeans$cluster, main="Slope", las=1,
  ylab="log(pen)", col=2:5, varwidth=TRUE)
boxplot(oxy ~ spe.kmeans$cluster, main="Oxygen", las=1,
  ylab="oxy", col=2:5, varwidth=TRUE)
boxplot(sqrt(amm) ~ spe.kmeans$cluster, main="Ammonium", las=1,
  ylab="sqrt(amm)", col=2:5, varwidth=TRUE)

# Test of ANOVA assumptions
# Normality of residuals
shapiro.test(resid(lm(sqrt(alt) ~
 as.factor(spe.kmeans$cluster))))
```

```
shapiro.test(resid(lm(log(pen) ~
 as.factor(spe.kmeans$cluster))))
shapiro.test(resid(lm(oxy ~ as.factor(spe.kmeans$cluster))))
shapiro.test(resid(lm(sqrt(amm) ~
 as.factor(spe.kmeans$cluster))))

 # Residuals of sqrt(alt), log(pen), oxy and sqrt(amm) are
 # normal, if the power of the test is adequate.
 # Try to find good normalizing transformations for
 # the other variables.

# Homogeneity of variances
bartlett.test(sqrt(alt), as.factor(spe.kmeans$cluster))
bartlett.test(log(pen), as.factor(spe.kmeans$cluster))
bartlett.test(oxy, as.factor(spe.kmeans$cluster))
bartlett.test(sqrt(amm), as.factor(spe.kmeans$cluster))

 # Variable sqrt(alt) has heterogeneous variances. It is not
 # appropriate for parametric ANOVA.

# ANOVA of the testable variables
summary(aov(log(pen) ~ as.factor(spe.kmeans$cluster)))
summary(aov(oxy ~ as.factor(spe.kmeans$cluster)))
summary(aov(sqrt(amm) ~ as.factor(spe.kmeans$cluster)))

 # Are slope, dissolved oxygen and dissolved ammonium
 # significantly different among species clusters?

# Kruskal-Wallis test of variable alt
kruskal.test(alt ~ as.factor(spe.kmeans$cluster))

 # Does altitude differ among clusters?

detach(env)
```

Hints *Note the use of* **`attach()`** *and* **`detach()`** *to avoid the repetition of the name of the object in all analyses.*

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.

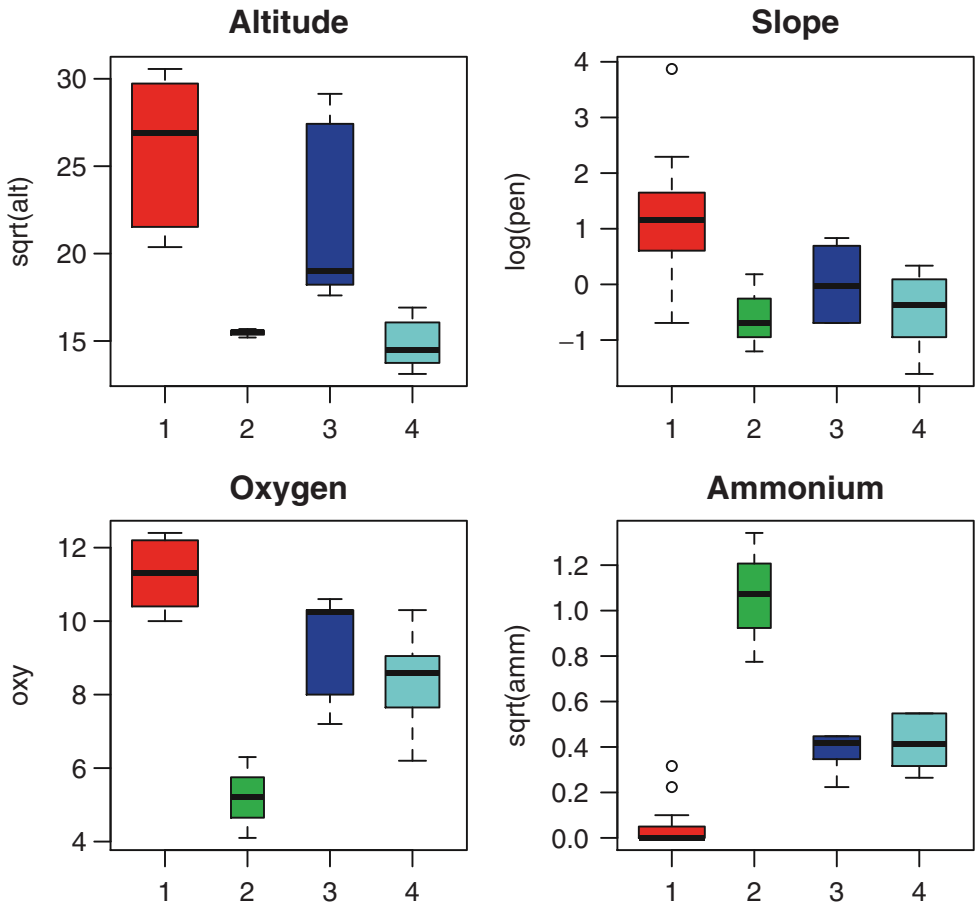

Fig. 4.17 Boxplots of four environmental variables grouped according to the four species-based *k*-means groups

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. 4.10.4). 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 are proposed in Chap. 6 to directly describe 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 chi-square test:

```
# Contingency table of two typologies
# **********************************

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

# Table crossing the k-means and environment 4-group typologies
table(as.factor(spe.kmeans$cluster),
as.factor(env.kmeans$cluster))
 # Are the two typologies telling the same story?

# Test the relationship using a chi-square test
chisq.test(as.factor(spe.kmeans$cluster),
 as.factor(env.kmeans$cluster))
# Change the testing procedure to a permutation test
chisq.test(as.factor(spe.kmeans$cluster),
 as.factor(env.kmeans$cluster),
 simulate.p.value=TRUE)
```

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 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.

```
# Mean abundances on k-means site clusters
# ****************************************

groups <- as.factor(spe.kmeans$cluster)
spe.means <- matrix(0, ncol(spe), length(levels(groups)))
row.names(spe.means) <- colnames(spe)
for(i in 1:ncol(spe)) {
  spe.means[i,] <- tapply(spe[,i], spe.kmeans$cluster, mean)
  }
# Mean species abundances of the four groups
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 make 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 χ^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 χ^2 test and has correct Type I error (Legendre 2010).

As a simple example, let us extract the most abundant species of the fish data set, classify them into several groups using *k*-means partitioning (Fig. 4.18), and run a global test (**`kendall.global()`**) to know if all groups of species (called "judges" in the original paper) are globally significantly associated. If it is the case, we shall run a posteriori tests (**`kendall.post()`**) on the species of each group to verify if all species within a group are concordant.

```
# Kendall's W coefficient of concordance
# ************************************

# Extraction of the most abundant species
sp.sum <- apply(spe, 2, sum)
spe.sorted <- spe[,order(sp.sum, decreasing=TRUE)]
spe.small <- spe.sorted[,1:20]

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

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

 # This result indicates that 2 groups may be a good choice.
 # Avoid solutions with groups containing a single species.

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

  # Now that we have two groups of species, let us run a global
  # Kendall W test on these groups.
spe.kendall.global <- kendall.global(spe.small.hel, clusters)
spe.kendall.global
```

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. 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 are not significant, it indicates that these groups have to be subdivided into smaller groups.

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

```
spe.kendall.post <- kendall.post(spe.small.hel, clusters,
nperm=9999)
spe.kendall.post
```

Look at the mean Spearman correlation coefficients. Within each group, concordant species must be positively associated with the others at a significant corrected *p*-value. Is it the case for all species? A species with a negative mean of its Spearman correlations with the other members of its cluster is incorrectly classified, suggesting that it should be left out of the group. If several species have negative mean Spearman correlations, the corresponding group should be split since not all members are belonging to the same association. Species whose corrected *p*-values are not significant are not contributing to the overall concordance of their group and should be left out.

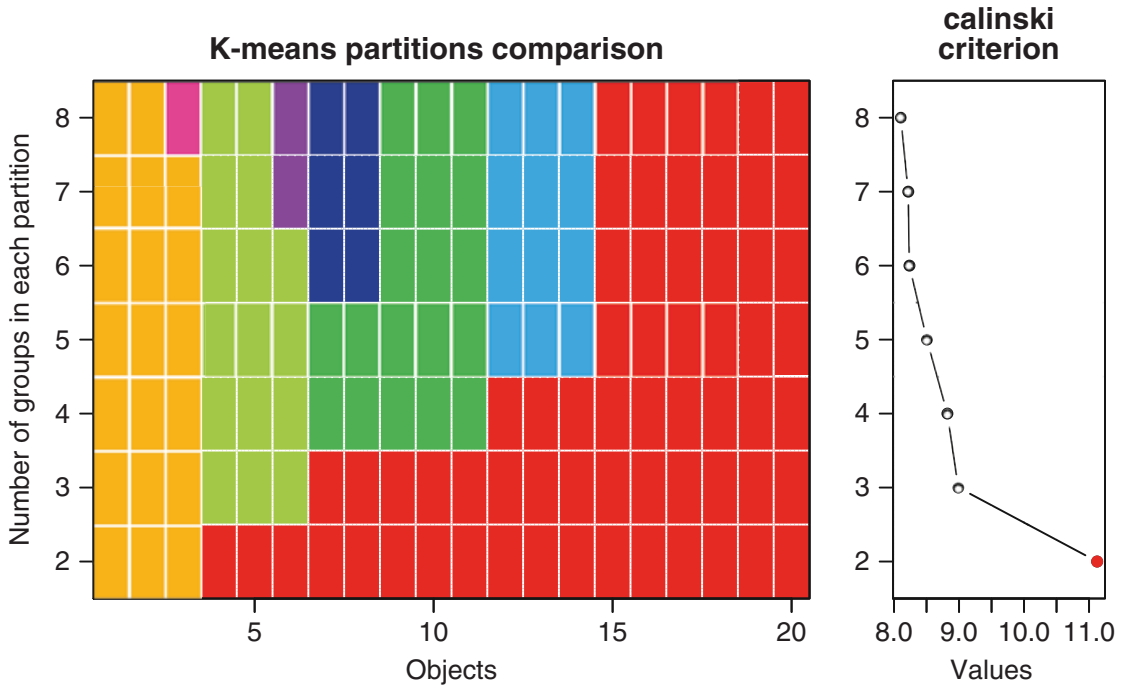

Fig. 4.18 *k*-means cascade plot showing the R-mode partitioning of the 20 most abundant fish species. The Calinski–Harabasz criterion points to an optimum of two groups

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 an RDA (Sect. 6.3) of the significantly associated species with respect to a set of explanatory environmental variables.

In this example, we ran the Kendall *W* analysis on the basis of the two groups suggested by the *k*-means partitioning of the species. A two-group partition is the most parsimonious model in this example, but that is not always the case. If the result had shown several misclassified species, we would have had either to reconsider the species typology (for instance, by trying other clustering methods), or rerun the analysis for three groups or more.

4.10.3 Species Assemblages in Presence–Absence Data

For presence–absence data, a new method is under development (Clua et al. 2010). The method consists in computing the *a* component of Jaccard's S_7 coefficient (as a measure of co-occurrences 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 distances: they have very small values for highly co-occurring species. We provide an **R** function called **`test.a()`** to compute this new coefficient. Readers are invited to apply it to the Doubs fish data transformed to presence–absences. A critical point is to specify enough permutations for the probabilities to survive a correction for multiple testing (see Sect. 7.2.6). In the Doubs data, there are 27 species, and thus $27 \times 26/2 = 351$ tests are run. A Bonferroni correction requires a *p*-value of $0.05/351 = 0.0001425$ to remain significant at the 0.05 level. This requires 99999 permutations, since the smallest *p*-value would be $1/(99999 + 1) = 0.00001$. Beware: this can be long on some computers.

```
# Species assemblages on presence-absence values
# *********************************************

source("test.a.R")   # Function must be in the working directory
spe.pa <- decostand(spe,"pa")       # Transform the data to
                                    # presence-absence
res <- test.a(spe.pa, nperm=99999)
```

```
summary(res)

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

res.p.vec <- as.vector(res$p.a.dist)
adjust.res <- p.adjust(res.p.vec,method="holm")

  # Among the corrected Holm p-values, find 0.05 or the closest
  # value smaller than 0.05. Identify the corresponding value in
  # the vector of uncorrected p-values.
  # In this case, it is 0.04878. There are around 83 p-values
  # smaller than 0.05 in the corrected Holm matrix (this number
  # could vary slightly among computer runs). The corresponding
  # uncorrected p-value is 0.00018.

  # The significant values thus have a probability of 0.00018 or
  # less. Replace all larger values in the dissimilarity matrix
  # by 1:

res.pa.dist <- res$p.a.dist
res.pa.dist[res.pa.dist>0.00018] <- 1

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

source("coldiss.R")  # Function must be in the working directory
coldiss(res.pa.dist, nc=10, byrank=TRUE, diag=TRUE)

  # One can also run a fuzzy non-hierarchical clustering (Section
  # 4.12). Try several k values to look for an optimal number of
  # clusters:

res.pa.fuz <- fanny(res.pa.dist, k=5, memb.exp=1.5)
summary(res.pa.fuz)
plot(silhouette(res.pa.fuz), main="Silhouette plot - Fuzzy
clustering", cex.names=0.8, col=res.pa.fuz$silinfo$widths+1)
res.pa.fuz$silinfo                # Silhouette information

  # The silhouette information consists in the most likely
  # membership to a "hard cluster" ("cluster"), the closest
  # neighbour cluster ("neighbor"),
  # and the silhouette width s(i) ("sil_width").
```

4.10.4 IndVal: Species Indicator Values

Dufrêne and Legendre (1997) proposed an original way to compute indicator values of species within clusters of sites. Their IndVal index combines a species mean abundances and its frequencies of occurrence in the groups. A high indicator value is obtained by a combination of large mean abundance within a group compared to the other groups (specificity) and presence in most sites of that group (fidelity). Dave Roberts, the author of the package **labdsv** used below, 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, albeit a little 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. Another approach, which is conceptually and statistically better, 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 *a posteriori* 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 of indicator species related to one explanatory variable, we could for instance ask ourselves if the variable `das` (distance from the source), acting as a surrogate of the overall river gradient, could be divided into groups upon which indicator species could be looked for.

```
# Species indicator values (Dufrene and Legendre)
# ***********************************************

# Divide the sites into 4 groups depending on the distance to
# the source of the river
das.D1 <- dist(data.frame(das=env[,1], row.names=rownames(env)))
dasD1.kmeans <- kmeans(das.D1, centers=4, nstart=100)
dasD1.kmeans$cluster

# Indicator species for this typology of the sites
(iva <- indval(spe, dasD1.kmeans$cluster))

 # The resulting object contains the following tables:
 #  - relfrq = relative frequency of the species in each group
```

```
  #  - relabu = relative abundance of the species across groups
  #  - indval = indicator value of each species
  # The two next items are extracted from the indval table
  # (group with the highest indval and indval value), and the
  # last contains the results of the permutation tests.

# Table of the significant indicator species
gr <- iva$maxcls[iva$pval<=0.05]
iv <- iva$indcls[iva$pval<=0.05]
pv <- iva$pval[iva$pval<=0.05]
fr <- apply(spe>0, 2, sum)[iva$pval<=0.05]
fidg <- data.frame(group=gr, indval=iv, pvalue=pv, freq=fr)
(fidg <- fidg[order(fidg$group, -fidg$indval),])
# Export the result to a CSV file (to be opened in a
# spreadsheet)
write.csv(fidg, "IndVal-das.csv")

  # On this basis, what do you think of the result? How does it
  # relate to the four ecological zones presented in Chapter 1?
```

In Sect. 4.11 you will find another application in relationship with the MRT method. Some detailed results are presented.

The search for indicator species has known some recent developments. De Cáceres and Legendre (2009) describe a series of 12 indices that can be used to identify indicator species. They belong to two groups: correlation indices, which should be used to determine the ecological preference of a given species among a set of alternative site groups, and indicator value indices, which are more useful for assessing the species predictive values, e.g. for field determination of community types or for ecological monitoring (De Cáceres and Legendre 2009, p. 3573). A new package **`indicspecies`**[1] computes different indicator species indices, including IndVal (or, actually, the square root of IndVal).

Two especially interesting features of this package are the computation of bootstrapped confidence intervals around indicator values (argument `nboot` of function **`strassoc()`**), and the (computer-intensive) possibility of pooling all groups of a typology in turn to look for species that may be indicators of pooled groups (function **`multipatt()`**).

[1] Available on http://www.bio.umontreal.ca/legendre/indexEn.html and http://sites.google.com/site/miqueldecaceres/software.

4.11 Multivariate Regression Trees: Constrained Clustering

4.11.1 Introduction

Multivariate regression trees (MRT; De'ath 2002) are an extension of univariate regression trees, a method allowing the recursive partitioning of a quantitative 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 rather focuses on prediction, making it a very interesting tool for practical applications, as in environmental management. The focus on prediction is imbedded in the method, as becomes clear below.

MRT is a powerful and robust method that 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 non-linear or high-order interactions among explanatory variables are present.

4.11.2 Computation (Principle)

The computation of an 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 see how they are applied together to produce a model that has the form of a decision tree.

4.11.2.1 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… $(n-1)$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 sum of squared

distances to the group mean (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. When that point is reached, 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 the RE; this would be equivalent to retain 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).*"

4.11.2.2 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 groups. A good predictive tree assigns the objects of the test set correctly, i.e. the true response variables of the objects newly assigned (on the basis of their explanatory variables) are close to the centroids of the response variables of the group where they are assigned (i.e. the predicted values). The performance of such a tree is assessed by its predictive error.

The measure of predictive error is CVRE. The function is:

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

where $y_{ij(k)}$ is one observation of the test set k, $\hat{y}_{j(k)}$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.

4.11.2.3 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 (by default $k = 10$) groups.
- 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). Formula (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 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 assignations of the objects into 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 MRT analysis, the computations of sums-of-squares 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.

4.11.3 Application Using Packages **mvpart** and **MVPARTwrap**

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.

Let us build a multivariate regression tree of the Doubs fish data constrained by the environmental variables. We 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 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 allow ourselves to interactively pick a solution on a graph provided by **mvpart()** (Fig. 4.19). The graph displays the relative error , which is steadily decreasing when the number of groups increases, and 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 which 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.

```
# Multivariate regression trees
# ****************************

spe.ch.mvpart <- mvpart(data.matrix(spe.norm) ~ ., env,
 margin=0.08, cp=0, xv="pick", xval=nrow(spe), xvmult=100,
 which=4)
# Here, click on the desired number of groups (for example 4)
summary(spe.ch.mvpart)
printcp(spe.ch.mvpart)
```

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

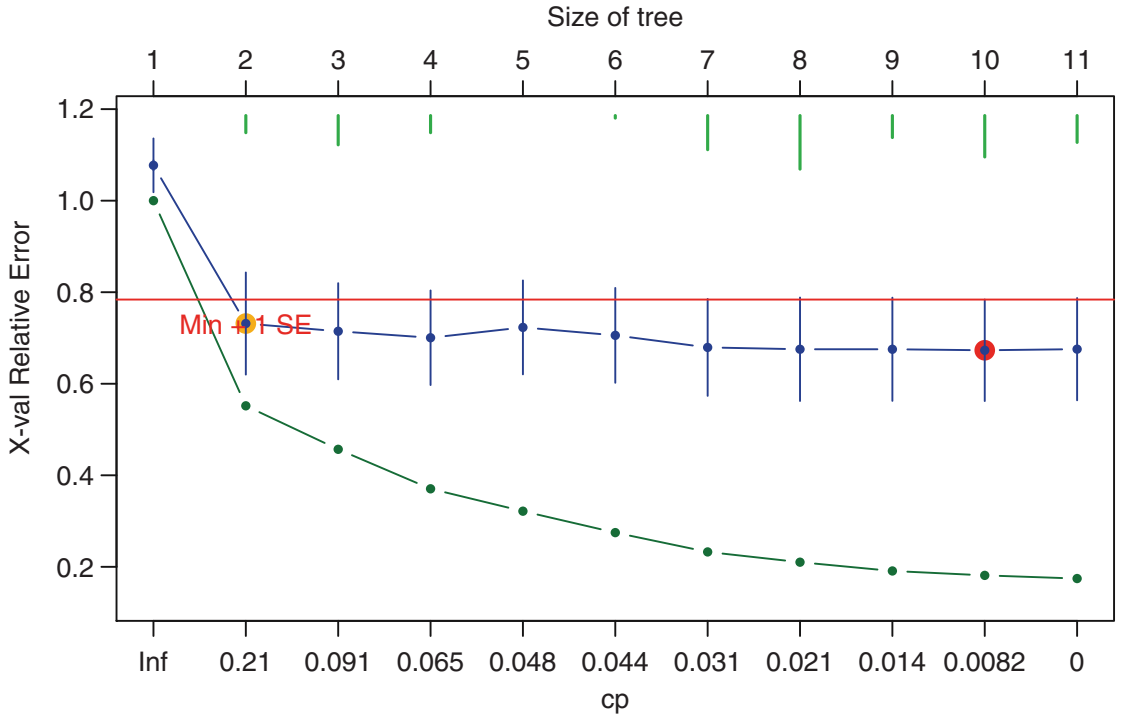

Fig. 4.19 Graph of the (steadily decreasing) relative error RE and the cross-validated relative error CVRE. The solution with the smallest CVRE is indicated (*red point*), as well as CVRE error bars. The *green bars* indicate the number of times that the solution was selected as the best during the cross-validation iterations

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 four-group solution. While not the absolute best, it still ranks among the good ones and avoids producing too many small groups (Fig. 4.20).

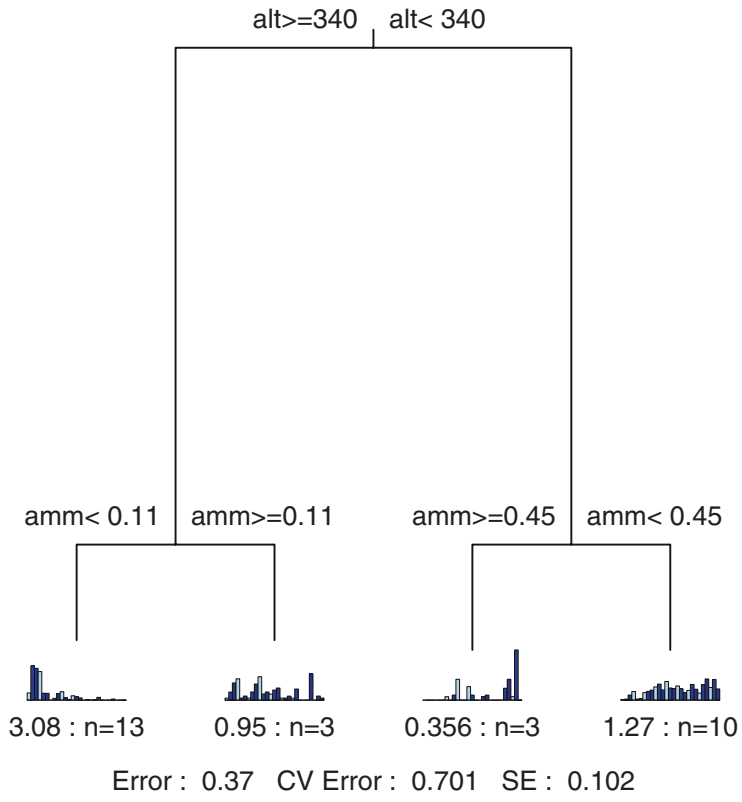

Fig. 4.20 Multivariate regression tree of the Doubs fish species explained by their environmental variables. Interpretation: see text

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 reciprocal 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 altitude. The critical value (here 340 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 `das` 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 bar plot 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 statistical approach is to search for characteristic or indicator species using **`indval()`** (Sect. 4.9). 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="grey")
plot(predict(spe.ch.mvpart), 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]),]

# 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=c("leaf #",
 groups.mrt[i]))
}
```

Unfortunately, extracting other numerical results from an **`mvpart()`** object is no easy task. This is why Marie-Hélène Ouellette wrote a wrapper doing just that and providing a wealth of additional, useful information. The package 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
# Packages MVPARTwrap and rdaTest must have been loaded
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 indeed 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 reciprocal of the tree RE. In our example, Fig. 4.20 gives an error of 0.37 for the tree, therefore the 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 TRU, VAI and LOC for the left branch (higher altitude), and ABL for the right one. The summary lists the sites present in each leaf. The result object itself contains the complete results from which the summary is drawn.

4.11.4 Combining MRT and IndVal

As suggested in the previous section, one could submit the MRT partition to a search for indicator species (IndVal, Sect. 4.10.4). This is better than visual examination of the results for the identification of "discriminant" species: one can test the significance of the indicator values.

```
# Indicator species search on the MRT result
spe.ch.MRT.indval <- indval(spe.norm, spe.ch.mvpart$where)
spe.ch.MRT.indval$pval    # Probability

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

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

Among other results, one finds here the permutation test results showing which species are significant indicators of the leaves. The following list gives the leaf number for the significant indicator species, followed by the list of the indicator values:

```
> spe.ch.MRT.indval$maxcls[which(spe.ch.MRT.indval$pval<=0.05)]
TRU VAI LOC HOT BAR SPI GOU BOU PSO CAR TAN BCO PCH GRE BBO ABL ANG
  1   1   1   4   4   4   4   4   4   4   4   4   4   4   4   3   4
>
> spe.ch.MRT.indval$indcls[which(spe.ch.MRT.indval$pval<=0.05)]
      TRU       VAI       LOC       HOT       BAR       SPI       GOU
0.7899792 0.5422453 0.4506428 0.5775629 0.6547659 0.5424497 0.4384538
      BOU       PSO       CAR       TAN       BCO       PCH       GRE
0.8743930 0.6568224 0.7964617 0.5304323 0.9000000 0.7000000 0.5632126
      BBO       ABL       ANG
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 (TRU), for instance, shows a significant indicator value of 0.7900 in the first leaf.

4.11.5 MRT as a "Chronological" Clustering Method

In cases where the data present themselves in a spatial or temporal sequence, the contiguity information can be taken into account when looking for groups (and 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. We suggest that MRT can be easily applied to such situations. The trick is simple: use a vector describing the sequence as the only explanatory variable. In our example data, the one-dimension position variable is `das` (distance from the source). One proceeds as previously. When interpreting the result, be aware that in the output object the site groups do not always follow the true order of the sequence. The result is displayed in Fig. 4.21.

```
# MRT as a constrained clustering method for spatial and
# temporal data sequences

spe.ch.seq <- mvpart(as.matrix(spe) ~ das, env, cp=0, xv="pick",
 margin=0.08, xval=nrow(spe), xvmult=100, which=4)
# Here, click on the desired number of groups

summary(spe.ch.seq)

# Group composition (labels of terminal nodes)
(gr <- spe.ch.seq$where)

# Renumber clusters sequentially
aa <- 1
gr2 <- rep(1,length(gr))
for (i in 2:length(gr)) {
  if (gr[i]!=gr[i-1]) aa <- aa+1
  gr2[i] <- aa
 }

# Plot of the clusters on a map of the Doubs river
plot(spa, asp=1, type="n", main="MRT groups",
 xlab="x coordinate (km)", ylab="y coordinate (km)")
lines(spa, col="light blue")
text(50, 10, "Upstream", cex=1.2, col="red")
text(25, 115, "Downstream", cex=1.2, col="red")
k <- length(levels(factor(gr2)))
for (i in 1:k) {
 points(spa[gr2==i,1], spa[gr2==i,2], pch=i+20, cex=3,
 col=i+1, bg=i+1)
 }
```

```
text(spa, row.names(spa), cex=0.8, col="white", font=2)
legend("bottomright", paste("Group",1:k), pch=(1:k)+20,
  col=2:(k+1), pt.bg=2:(k+1), pt.cex=2, bty="n")
```

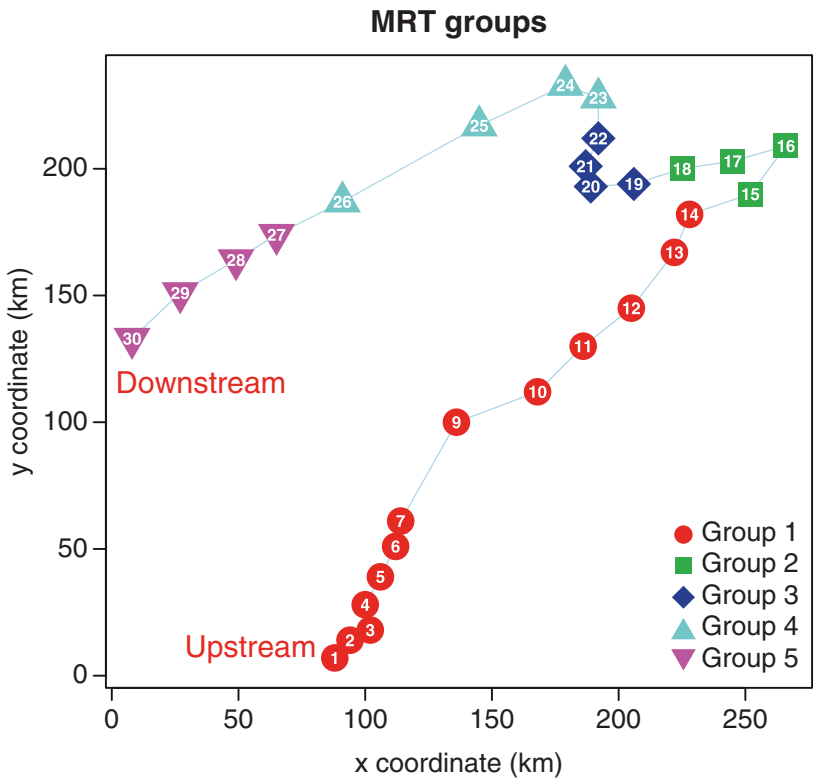

Fig. 4.21 Clustering with contiguity constraint using MRT

4.12 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 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. Consequently, a different family of hierarchical and non-hierarchical methods has been developed, namely, fuzzy clustering. We do 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 1990).

4.12.1 Fuzzy c-Means Clustering Using `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.

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 distance matrices. In the former case, the default metric is `euclidean`. Here, we directly use as input the chord distance matrix previously computed from the fish species data. An equivalent result would be obtained by running the chord-transformed species data "`spe.norm`" with the `metric="euclidean"` option.

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. 4.22) and we replace the original ordination diagram by a PCoA combined with star plots of the objects (Fig. 4.23). Each object is associated with a small "star" whose segment radiuses are proportional to its membership coefficient.

```
# Fuzzy c-means clustering of the fish species data
# ************************************************

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

```
spefuz.g <- spe.fuz$clustering

# Site membership
spe.fuz$membership

# Nearest crisp clustering
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)
dc.pcoa <- cmdscale(spe.ch)
dc.scores <- scores(dc.pcoa, choices=c(1,2))

# Ordination plot of fuzzy clustering result
plot(scores(dc.pcoa), asp=1, type="n",
 main="Ordination of fuzzy clusters (PCoA)")
abline(h=0, lty="dotted")
abline(v=0, lty="dotted")

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=scores(dc.pcoa),
 draw.segments=TRUE, add=TRUE, scale=FALSE, len=0.1,
 col.segments=2:(k+1))
legend(locator(1), paste("Cluster", 1:k, sep=" "),
 pch=15, pt.cex=2, col=2:(k+1), bty="n")

# Click on the graph to position legend
```

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

In the **`legend()`** *function, argument* `locator(1)` *allows users to position the legend on the graph interactively, by clicking on the desired position.*

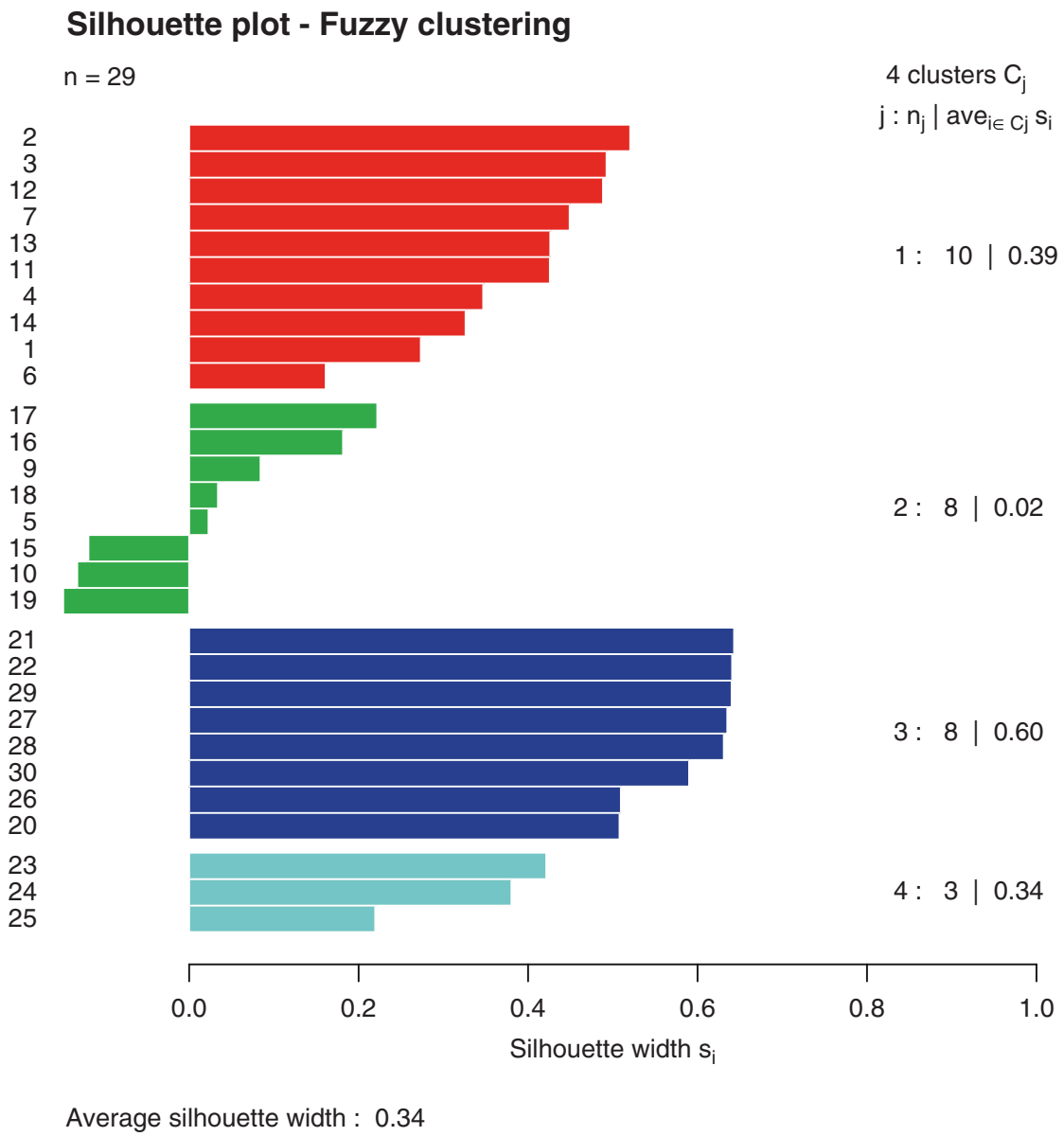

Fig. 4.22 Silhouette plot of the *c*-means fuzzy clustering of the fish data preserving the chord distance

The silhouette plot (Fig. 4.22) shows, in particular, that cluster 2 is not well defined. On the ordination diagram (Fig. 4.23), the star plots of the ill-classified objects (10, 15, 19) also illustrate that their membership is unclear.

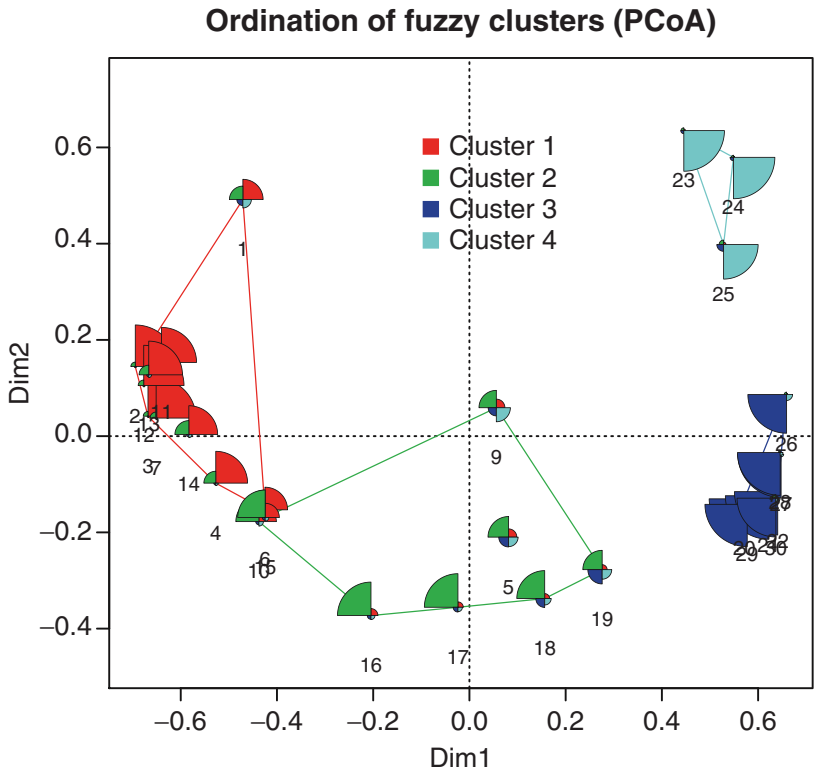

Fig. 4.23 *c*-Means fuzzy clustering of the fish data preserving the chord distance. Principal coordinate ordination associated with star plots showing the memberships of the sites. Cluster shadings as in Fig. 4.22

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.

While "hard" clustering may appear somewhat unrealistic in ecology, its application is of great help as soon as one needs a typology or a decision-making tool requiring unambiguous allocation of sites. Fuzzy clustering is a more nuanced, and therefore more realistic approach in most ecological situations if its purpose is to describe 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.

4.13 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 very 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 optimize the output of your research.

Chapter 5
Unconstrained Ordination

5.1 Objectives

While cluster analysis looks for discontinuities in a dataset, ordination extracts the main trends in the form of continuous axes. It is therefore particularly well adapted to analyse data from natural ecological communities, which are generally structured in gradients.

Practically, you will:

- Learn how to choose among various ordination techniques (PCA, CA, PCoA and NMDS), compute them using the correct options, and properly interpret the ordination diagrams
- Apply these techniques to the Doubs river data
- Overlay the result of a cluster analysis on an ordination diagram to improve the interpretation of the clustering results
- Interpret the structures in the species data using the environmental variables from a second dataset
- Write your own PCA function

5.2 Ordination Overview

5.2.1 Multidimensional Space

A multivariate data set can be viewed as a collection of sites positioned in a space where each variable defines one dimension. There are thus as many dimensions as variables. To reveal the structure of the data, it would be interesting to represent the main trends in the form of scatterplots of the sites. Since ecological data generally contain more than two variables, it is tedious and not very informative to draw the objects in a series of planes defined by all possible pairs of descriptors. For instance, if the matrix contains ten descriptors, the number of planes to draw would

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_5, © Springer Science+Business Media, LLC 2011

be equal to $(10 \times 9)/2 = 45$. Such a series of scatterplots would allow neither to bring out the most important structures of the data, nor to visualize the relationships among descriptors (which, in general, are not linearly independent of one another). The aim of ordination methods is to represent the data along a reduced number of orthogonal axes, constructed in such a way that they represent, in decreasing order, the main trends of the data. These trends can then be interpreted visually or in association with other methods such as clustering or regression. Here, we shall address four basic techniques. All these methods are descriptive: no statistical test is provided to assess the significance of the structures detected. That is the role of constrained ordination, a family of methods that are presented in Chap. 6.

5.2.2 Ordination in Reduced Space

Most ordination methods (except NMDS) are based on the extraction of the eigenvectors of an association matrix. They can be classified according to the distance preserved among sites and to the type of variables that they can handle. Legendre and Legendre (1998, Table 9.1, p. 388) provide a table showing their domains of application.

The basic principle of ordination in reduced space is the following. Imagine an $n \times p$ data set containing n objects and p variables. The n objects can be represented as a cluster of points in the p-dimensional space. Now, this cluster is generally not spheroid: it is elongated in some directions and flattened in others. These directions are not necessarily aligned with a single dimension (= a single variable) of the multidimensional space. The direction where the cluster is most elongated corresponds to the direction of largest variance of the cluster. This is the first axis that an ordination will extract. The next axis to be extracted is the second most important in variance, provided that it is *orthogonal* (linearly independent, uncorrelated) to the first one. The process continues until all axes have been computed.

When there are a few major structures in the data (gradients or groups) and the method has been efficient at extracting them, then the few first axes contain most of the useful information, i.e. they have extracted most of the variance of the data. In that case, the distances among sites in the projection in reduced space (most often two-dimensional) are relatively similar to the distances among objects in the multidimensional space. Note, however, that an ordination can be useful even when the first axes account for small proportions of the variance. This may happen when there are some interesting structures in an otherwise noisy data set. The question arising is then: how many axes should one retain and interpret? In other words, how many axes represent interpretable structures? The answer depends on the method; several helping procedures are explained in due course to answer this question.

The methods that are presented in this chapter are:

- *Principal component analysis (PCA)*: the main eigenvector-based method. Works on raw, quantitative data. Preserves the Euclidean distance among sites.
- *Correspondence analysis (CA)*: works on data that must be frequencies or frequency-like, dimensionally homogeneous, and non-negative. Preserves the χ^2 distance among rows or columns. Mainly used in ecology to analyse species data tables.
- *Principal coordinate analysis (PCoA)*: devoted to the ordination of distance matrices, most often in the Q mode, instead of site-by-variables tables. Hence, great flexibility in the choice of association measures.
- *Nonmetric multidimensional scaling (NMDS)*: unlike the three others, this is not an eigenvector-based method. NMDS tries to represent the set of objects along a predetermined number of axes while preserving the ordering relationships among them.

PCoA and NMDS can produce ordinations from any square distance matrix.

5.3 Principal Component Analysis

5.3.1 Overview

Imagine a data set whose variables are normally distributed. This data set is said to show a multinormal distribution. The first principal axis (or principal-component axis) of a PCA of this data set is the line that goes through the greatest dimension of the concentration ellipsoid describing this multinormal distribution. The following axes, which are orthogonal to one another and successively shorter, go through the following greatest dimensions of the ellipsoid (Legendre and Legendre 1998). One can derive a maximum of p principal axes from a data set containing p variables.

Stated otherwise, PCA carries out a rotation of the original system of axes defined by the variables, such that the successive new axes (called principal components) are orthogonal to one another, and correspond to the successive dimensions of maximum variance of the scatter of points. The principal components give the positions of the objects in the new system of coordinates. PCA works on a *dispersion matrix* $\boldsymbol{S}$, i.e. an association matrix among variables containing the variances and covariances of the variables, or the correlations computed from dimensionally heterogeneous variables. It is exclusively devoted to the analysis of quantitative variables. The distance preserved is the Euclidean distance and the relationships detected are linear. Therefore, it is not generally appropriate to the

analysis of raw species abundance data. These can, however, be subjected to PCA after an appropriate pre-transformation (Sects. 2.2.4 and 5.3.3).

In a PCA ordination diagram, following the tradition of scatter diagrams in Cartesian coordinate systems, *objects* are represented as *points* and *variables* are displayed as *arrows.*

Later in this chapter, we show how to program a PCA in **R** using matrix equations. But for everyday users, PCA is available in several **R** packages. A convenient function for ecologists is **`rda()`** in package **`vegan`**. The name of the function refers to redundancy analysis, a method that is presented in Chap. 6. Other possible functions (not detailed here) are **`dudi.pca()`** (package **`ade4`**) and **`prcomp()`** (package **`stats`**).

5.3.2 PCA on the Environmental Variables of the Doubs Data Set Using `rda()`

Let us work again on the Doubs data. We have 11 quantitative environmental variables at our disposal. How are they correlated? What can we learn from the ordination of the sites?

Since the variables are expressed in different measurement scales, we compute a PCA on the correlation matrix. Correlations are the covariances of standardized variables.

5.3.2.1 Preparation of the Data

```
# Load required  packages
library(ade4)
library(vegan)
library(gclus)
library(ape)

# Import the data from CSV files
spe <- read.csv("DoubsSpe.csv", row.names=1)
env <- read.csv("DoubsEnv.csv", row.names=1)
spa <- read.csv("DoubsSpa.csv", row.names=1)
# Remove empty site 8
spe <- spe[-8,]
env <- env[-8,]
spa <- spa[-8,]
# A reminder of the content of the env dataset
summary(env) # Descriptive statistics
```

5.3.2.2 PCA on a Correlation Matrix

```
# PCA on the full dataset (correlation matrix: scale=TRUE)
# *******************************************************

env.pca <- rda(env, scale=TRUE)  # Argument scale=TRUE calls for
                                 # a standardization of the
                                 # variables
env.pca
summary(env.pca) # Default scaling 2
summary(env.pca, scaling=1)

  # Note that the scaling (see below) is called at the step of
  # the summary (or, below, for the drawing of biplots) and not
  # for the analysis itself.
```

The "summary" output is presented as follows for scaling 2 (some results have been deleted):

```
Call:
rda(X = env, scale = TRUE)

Partitioning of correlations:
              Inertia Proportion
Total              11          1
Unconstrained      11          1

Eigenvalues, and their contribution to the correlations

Importance of components:
                         PC1   PC2    PC3   PC4   PC5   PC6...
Eigenvalue             6.098 2.167 1.0376 0.704 0.352 0.319...
Proportion Explained   0.554 0.197 0.0943 0.064 0.032 0.029...
Cumulative Proportion  0.554 0.751 0.8457 0.910 0.942 0.971...

Scaling 2 for species and site scores
* Species are scaled proportional to eigenvalues
* Sites are unscaled: weighted dispersion equal on all
* dimensions
* General scaling constant of scores:  4.189264
```

```
Species scores

          PC1      PC2       PC3       PC4      PC5       PC6
das  1.08432  0.5148 -0.257430 -0.16170  0.21140 -0.09500
alt -1.04356 -0.5946  0.179904  0.12274  0.12527  0.14024
(...)

Site scores (weighted sums of species scores)

         PC1       PC2       PC3       PC4     PC5       PC6
1  -1.41239 -1.47577 -1.74581 -2.95537  0.2312  0.49150
2  -1.04170 -0.81766  0.34078  0.54374  0.9252 -1.77040
```

The ordination output uses some vocabulary that requires explanations.

- *Inertia*: in **vegan**'s language, this is the general term for "variation" in the data. This term comes from the world of CA (Sect. 5.4). In PCA, the "inertia" is either the sum of the variances of the variables (PCA on a covariance matrix) or, as in this case (PCA on a correlation matrix), the sum of the diagonal values of the correlation matrix, i.e. the sum of all correlations of the variables with themselves, which corresponds to the number of variables (11 in this example).
- *Constrained and unconstrained*: see Chap. 6 (canonical ordination). In PCA, the analysis is unconstrained, and so are the results.
- *Eigenvalues*: symbolized λ_j, these are measures of the importance (variance) of the axes. They can be expressed as *Proportions Explained*, or proportions of variation accounted for, by dividing them by the total inertia.
- *Scaling*: not to be confused with the argument `scale` calling for standardization of variables. "Scaling" refers to the way ordination results are projected in the reduced space for graphical display. There is no single way to optimally display objects and variables together in a PCA biplot, i.e. a plot showing two types of results, here the sites and the variables. Two main types of scaling are generally used. Each of them has properties that must be kept in mind for proper interpretation of the biplots. Here, we give the essential features of each scaling. Please refer to Legendre and Legendre (1998, pp. 403–404) for a complete account.
  - *Scaling 1* = distance biplot: the eigenvectors are scaled to unit length. (1) **Distances among objects in the biplot are approximations of their Euclidean distances in multidimensional space**. (2) *The angles among descriptor vectors are meaningless.*

  - *Scaling 2* = correlation biplot: each eigenvector is scaled to the square root of its eigenvalue. (1) *Distances among objects in the biplot are not approximations of their Euclidean distances in multidimensional space.* (2) **The angles between descriptors in the biplot reflect their correlations**.
  - In both cases, projecting an object at right angle on a descriptor approximates the position of the object along that descriptor.
  - Bottom line: if the main interest of the analysis is to interpret the relationships among **objects**, choose **scaling 1**. If the main interest focuses on the relationships among **descriptors**, choose **scaling 2**.

- *Species scores*: coordinates of the arrow heads of the variables. For historical reasons, response variables are always called "species" in **`vegan`**, no matter what they represent.
- *Site scores*: coordinates of the sites in the ordination diagram. Objects are always called "Sites" in **`vegan`** output files.

5.3.2.3 Extracting, Interpreting and Plotting Results from a `vegan` Ordination Output Object

`vegan` output objects are complex entities, and extraction of their elements does not follow the basic rules of **R**. Type **`?cca.object`** in the **R** console. This calls for a help file explaining all features of an **`rda()`** or **`cca()`** output object. The examples at the end of that help file show how to access some of the ordination results directly. Here, we access some important results as examples. Further results are examined later when useful.

Eigenvalues

First, let us examine the eigenvalues. Are the first few clearly larger than the following ones? Here, a question arises: how many ordination axes are meaningful to display and interpret?

PCA is not a statistical test, but a heuristic procedure: it aims at representing the major features of the data along a reduced number of axes (hence, the expression "ordination in reduced space"). Usually, the user examines the eigenvalues, and decides how many axes are worth representing and displaying on the basis of the amount of variance explained. The decision can be completely arbitrary (for instance, interpret the number of axes necessary to represent 75% of the variance of the data), or assisted by one of several procedures proposed to set a limit between the axes that represent interesting variation of the data and axes that merely display the remaining, essentially random variance. One of these procedures

(called the Kaiser–Guttman criterion) consists in computing the *mean of all eigenvalues* and interpreting only the axes whose eigenvalues are larger than that mean. Another is to compute a *broken stick model*, which randomly divides a stick of unit length into the same number of pieces as there are PCA axes. The theoretical equation for the broken stick model is known. The pieces are then put in order of decreasing length and compared to the eigenvalues. One interprets only the axes whose eigenvalues are larger than the length of the corresponding piece of the stick, or, alternately, one may compare the sum of eigenvalues, from 1 to k, to the sum of the values from 1 to k predicted by the broken stick model. One can compute these two procedures by hand as follows (Fig. 5.1)[1]:

```
# Examine and plot partial results from PCA output
# ************************************************

?cca.object  # Explains how an ordination object produced by
             # vegan is structured and how to extract its
             # results.

# Eigenvalues
(ev <- env.pca$CA$eig)

# Apply Kaiser-Guttman criterion to select axes
ev[ev > mean(ev)]

# Broken stick model
n <- length(ev)
bsm <- data.frame(j=seq(1:n), p=0)
bsm$p[1] <- 1/n
for (i in 2:n) {
 bsm$p[i] = bsm$p[i-1] + (1/(n + 1 - i))
}
bsm$p <- 100*bsm$p/n
bsm

# Plot eigenvalues and % of variance for each axis
par(mfrow=c(2,1))
barplot(ev, main="Eigenvalues", col="bisque", las=2)
abline(h=mean(ev), col="red")    # average eigenvalue
legend("topright", "Average eigenvalue", lwd=1, col=2, bty="n")
barplot(t(cbind(100*ev/sum(ev),bsm$p[n:1])), beside=TRUE,
 main="% variance", col=c("bisque",2), las=2)
```

[1] Comparison of a PCA result with the broken stick model can also be done by using function **`PCAsignificance()`** of package **`BiodiversityR`**.

```
legend("topright", c("% eigenvalue", "Broken stick model"),
 pch=15, col=c("bisque",2), bty="n")

 # Is the same number of axes retained by the two rules?
```

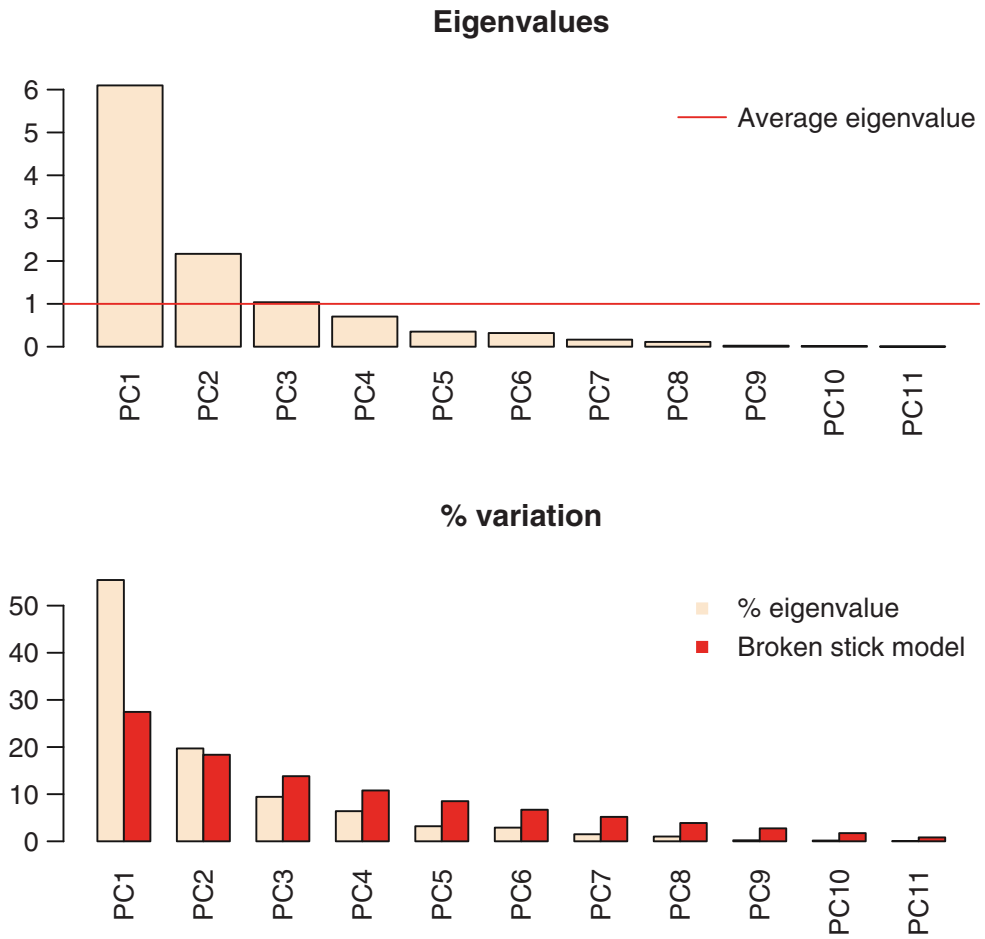

Fig. 5.1 Kaiser–Guttman and broken-stick plots to help assess the number of interpretable axes in PCA. Application to the Doubs environmental data

To make things easier, the code above has been framed in a function called **evplot()**, which is used as follows:

```
# Same plots using a single function:
# Plot eigenvalues and % of variance for each axis
source("evplot.R")
evplot(ev)
```

Biplots of Sites and Variables

To plot PCA results in a proper manner, one has to show objects as points and variables as arrows. Two plots are produced here, the first in scaling 1 (optimal display of objects), the second in scaling 2 (optimal display of variables) (Fig. 5.2). We present two functions: **vegan**'s **biplot.rda()** and a function directly drawing scaling 1 and 2 biplots from **vegan** results: **cleanplot.pca()**.

```
# Two PCA biplots: scaling 1 and scaling 2
# ****************************************

# Plots using biplot.rda
par(mfrow=c(1,2))
biplot.rda(env.pca, scaling=1, main="PCA - scaling 1")
biplot.rda(env.pca, main="PCA - scaling 2")  # Default scaling 2

# Plots using cleanplot.pca
# A rectangular graphic window is needed for the two plots
source("cleanplot.pca.R")
# With points for sites and arrowheads
cleanplot.pca(env.pca, point=TRUE)
# With site labels only (vegan's standard)
cleanplot.pca(env.pca)
cleanplot.pca(env.pca, ahead=0)   # ... and without arrowheads

 # What does the circle in the left-hand plot mean? See below...
```

Hint *Check in the* **cleanplot.pca()** *function how the plots are progressively built. First, one extracts two data tables (scaling 1 and 2, site scores and species scores) from the rda output object by means of the* **scores()** *function. Then an empty plot is produced using a special* **vegan** *plotting function called* **plot.cca()**. *Site points (function* **points()**) *and site labels (function* **text()**) *are added afterwards. Finally, species arrows and their labels are drawn, and a circle (see below) is added to the scaling 1 biplot.*

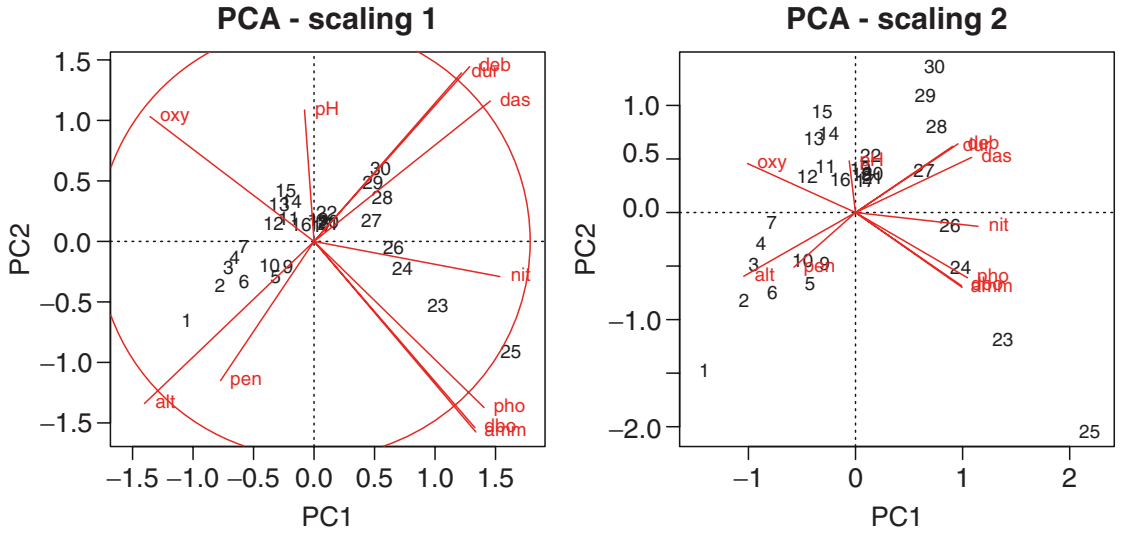

Fig. 5.2 PCA biplots of the Doubs environmental data, drawn with function **`cleanplot.pca()`**

Now, it is time to interpret the two biplots. First, the proportion of variance accounted for by the first two axes is 0.751 or 75.1%. This high value makes us confident that our interpretation of the first pair of axes extracts most relevant information from the data. Here is an example of how such a biplot can be interpreted.

First, the *scaling 1 biplot* displays a feature that must be explained. The circle is called a *circle of equilibrium contribution*. Its radius is equal to $\sqrt{d / p}$, where d is the number of axes represented in the biplot (usually $d=2$) and p is the number of dimensions of the PCA space (i.e. usually the number of variables of the data matrix).[2] The radius of this circle represents the length of the vector representing a variable that would contribute equally to all the dimensions of the PCA space. Therefore, for any given pair of axes, the variables that have vectors longer than this radius make a higher contribution than average and can be interpreted with confidence.

The *scaling 1 biplot* shows a gradient from left to right, starting with a group formed by sites 1–10 which display the highest values of altitude (`alt`) and slope (`pen`), and the lowest values in river discharge (`deb`) and distance from the source (`das`); hardness (`dur`), which increases in the downstream direction, is also correlated

[2] Note, however, that **`vegan`** uses an internal constant to rescale its results, so that the vectors and the circle represented here are not equal but proportional to their original values. See the code of the **`cleanplot.pca()`** function.

to these variables. The second group of sites (11–16), has the highest values in oxygen content (`oxy`) and the lowest in nitrate concentration (`nit`). A third group of very similar sites (17–22) shows intermediate values in almost all the measured variables; they are not spread out by the variables contributing to axes 1 and 2. Phosphate (`pho`) and ammonium (`amm`) concentrations, as well as biological oxygen demand (`dbo`) show their maximum values around sites 23–25; the values decrease afterwards. Overall, the progression from oligotrophic, oxygen-rich to eutrophic, oxygen-deprived water is clear.

The *scaling 2 biplot* shows that the variables are organized in groups. The lower left part of the biplot shows that altitude and slope are very highly, positively correlated, and that these two variables are very highly, negatively correlated with another group comprising distance from the source, river discharge and calcium concentration. Oxygen content is positively correlated with slope (`pen`) and altitude, but very negatively with phosphate and ammonium concentration and, of course, with biological oxygen demand. The right part of the diagram shows the variables associated with the lower section of the river, i.e. the group discharge (`deb`) and hardness (`dur`), highly correlated with the distance from the source, and the group of variables linked to eutrophication, i.e. phosphate and ammonium concentration and biological oxygen demand. Positively correlated with these two groups is nitrate concentration (`nit`). Nitrate and pH have nearly orthogonal arrows, indicating a correlation close to 0. pH displays a shorter arrow, showing its lesser importance for the ordination of the sites in the ordination plane. A plot of axes 1 and 3 would emphasize its contribution to axis 3.

This example shows how useful a biplot representation can be in summarizing the main features of a data set. Clusters and gradients of sites are obvious, as are the correlations among the variables. The correlation biplot (scaling 2) is far more informative than the visual examination of a correlation matrix among variables; the latter can be obtained by typing **`cor(env)`**.

Technical remark: **`vegan`** provides a simple plotting function for ordination results, called **`plot.cca()`**. However, the basic use of this function provides PCA plots where sites as well as variables are represented by points. This is misleading, since the points representing the variables are actually the apices (tips) of vectors that must be drawn for the plot to be interpreted correctly.

Supplementary sites and species can be added to a PCA plot through the function **`predict.cca()`**. Explanatory variables can be added through the function **`envfit()`**.

5.3.2.4 Combining Clustering and Ordination Results

Comparing a cluster analysis and an ordination can be fruitful to explain or confirm the differences between groups of sites. Here, you will see two ways of combining these results. The first differentiates clusters of sites by colours on the ordination plot, the second overlays a dendrogram on the plot. Both are done on a single PCA plot here (Fig. 5.3), but they can be drawn separately, of course.

```
# Combining clustering and ordination results
# ***************************************

# Clustering the objects using the environmental data: Euclidean
# distance after standardizing the variables, followed by Ward
# clustering

env.w <- hclust(dist(scale(env)), "ward")

# Cut the dendrogram to yield 4 groups
gr <- cutree(env.w, k=4)
grl <- levels(factor(gr))

# Get the site scores, scaling 1
sit.sc1 <- scores(env.pca, display="wa", scaling=1)

# Plot the sites with cluster symbols and colours (scaling 1)
p <- plot(env.pca, display="wa", scaling=1, type="n",
 main="PCA correlation + clusters")
abline(v=0, lty="dotted")
abline(h=0, lty="dotted")
for (i in 1:length(grl)) {
 points(sit.sc1[gr==i,], pch=(14+i), cex=2, col=i+1)
 }
text(sit.sc1, row.names(env), cex=.7, pos=3)

# Add the dendrogram
ordicluster(p, env.w, col="dark grey")
legend(locator(1), paste("Group",c(1:length(grl))),
 pch=14+c(1:length(grl)),
 col=1+c(1:length(grl)), pt.cex=2)
```

Hint *See how the coding of the symbols and colours is conditioned on the number of groups automatically: object* `grl` *has been set to contain numbers from 1 to the number of groups.*

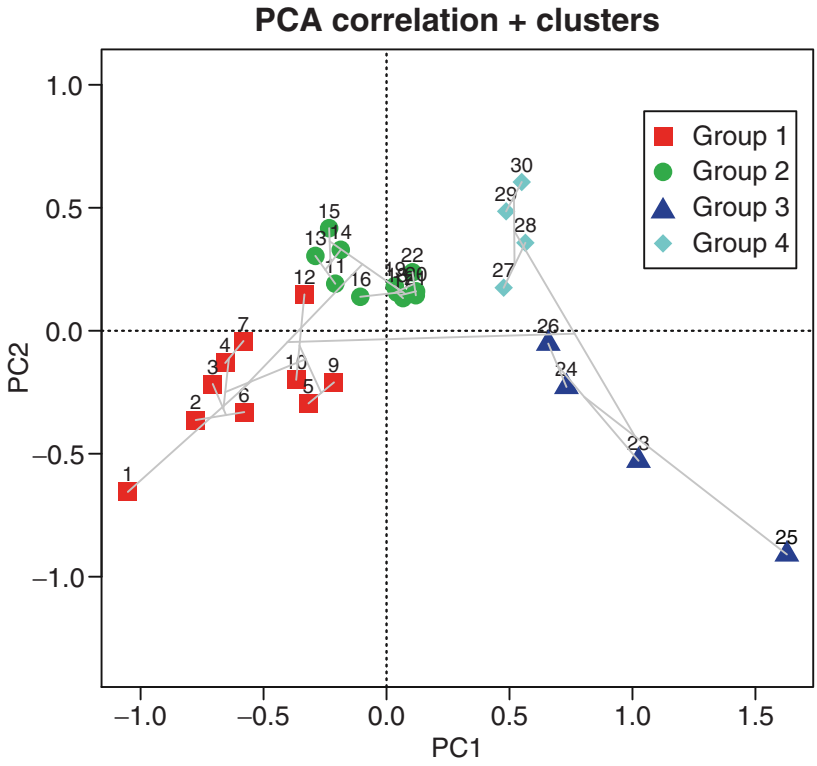

Fig. 5.3 PCA biplot (scaling 1) of the Doubs environmental data with overlaid clustering results

5.3.3 PCA on Transformed Species Data

PCA being a linear method preserving the Euclidean distance among sites, it is not naturally adapted to the analysis of species abundance data. However, transforming these after Legendre and Gallagher (2001) alleviates this problem (Sect. 2.2.4). Here is a quick application with a Hellinger pre-transformation on the fish data (Fig. 5.4).

```
# PCA on the fish abundance data
# ******************************

# Hellinger pre-transformation of the species data
spe.h <- decostand(spe, "hellinger")
spe.h.pca <- rda(spe.h)
spe.h.pca

# Plot eigenvalues and % of variance for each axis
ev <- spe.h.pca$CA$eig
evplot(ev)

# PCA biplots
cleanplot.pca(spe.h.pca, ahead=0)

  # The species do not form clear groups like the environmental
  # variables. However, see how the species replace one another
  # along the site sequence.
  # On the scaling 1 biplot, observe that 8 species contribute
  # strongly to axes 1 and 2. Are these species partly of
  # completely the same as those identified as indicator of the
  # groups in Section 4.10.3?
```

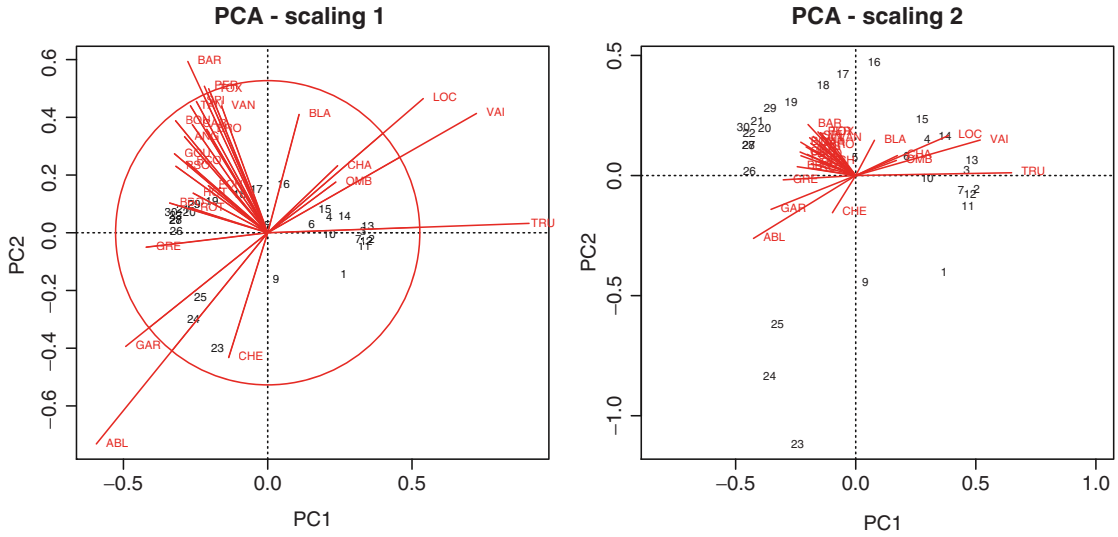

Fig. 5.4 PCA biplots of the Hellinger-transformed fish species data

For comparison, repeat the PCA on the original file `spe` without transformation. Which ordination shows better the gradient of species contributions along the course of the river?

Although PCA has a long history as a method devoted to tables of physical and chemical variables, the recent introduction of species data pre-transformations has opened up this powerful technique to the analysis of community data. Although PCA itself is not modified and remains a linear ordination model, the pre-transformations ensure that the species data are treated according to their specificity, i.e. without undue importance being given to double zeros. A scaling 1 PCA biplot thus reveals the underlying gradients structuring the community; the sites are ordered along the axes according to their positions along these gradients. The circle of equilibrium contribution allows the identification of the species contributing most to the plotted pair of axes. A scaling 2 biplot reveals the relationships among species in a correlation-like fashion; since the data have been transformed, the correlations are not completely equivalent to Pearson's *r* computed on raw data.

The chi-square transformation can also be applied to species data prior to PCA. In that case, the PCA solution is very similar, but not identical to a CA of the species data. Although the two methods preserve the chi-square distance among the sites, the calculation of the eigen-decomposition is not done in exactly the same way and leads to different sets of eigenvalues and eigenvectors.

5.3.4 Domain of Application of PCA

Principal component analysis is a very powerful technique, but it has its limits. The main application of PCA in ecology is the ordination of sites on the basis of quantitative environmental variables or, after an appropriate transformation, of community composition data. PCA has originally been defined for data with multinormal distributions. In its applications in ecology, however, PCA is not very sensitive to departure from multinormality, as long as the distributions are not exaggeratedly skewed. The main computational step of PCA is the eigen-decomposition of a dispersion matrix (linear covariances or correlations). Covariances must in turn be computed on quantitative data – but see below for binary data. Here are, in more detail, the conditions of application of PCA:

- PCA must be computed on a table of dimensionally homogeneous variables. The reason is that it is the sum of the variances of the variables that is partitioned into eigenvalues. Variables must be in the same physical units to produce a meaningful sum of variances (the units of a variance is the square of the units of the variable from which it was computed), or they must be dimensionless, which is the case for standardized or log-transformed variables.

- The data matrix must not be transposed since covariances or correlations among objects are meaningless.
- Covariances and correlations are defined for quantitative variables. However, PCA is very robust to variations in the precision of data. Since a Pearson correlation coefficient on semi-quantitative data is equivalent to a Spearman's correlation, a PCA on such variables yields an ordination where the relationship among variables is estimated using that measure.
- PCA can be applied to binary data. Gower (1966, *in* Legendre and Legendre 1998) has shown that, with binary descriptors, PCA positions the objects, in the multidimensional space, at distances that are the square roots of complements of simple matching coefficients S_1 (i.e. $\sqrt{1-S_1}$) times a constant which is the square root of the number of binary variables.
- Species presence–absence data can be subjected to a Hellinger or chord transformation prior to PCA. The justification is that the Hellinger and chord distances computed on presence–absence data are both equal to $\sqrt{2}\sqrt{1-\text{Ochiai similarity}}$, so PCA after Hellinger or chord transformation preserves the Ochiai distance among objects in scaling type 1 plots. We also know that $\sqrt{1-\text{Ochiai similarity}}$ is a metric distance (Legendre and Legendre 1998, Table 7.2) which is appropriate for the analysis of community composition presence–absence data.
- Avoid the mistake of interpreting the relationships among variables based on the proximities of the apices (tips) of the vector arrows instead of their angles in biplots.

5.3.5 PCA Using Function `PCA()`

For someone who wants a quick assessment of the structure of his or her data, a quick way is to use functions **`PCA()`** and **`biplot.PCA()`**. Here is how they work (example on the Doubs environmental data).

```
# PCA on the environmental data set using PCA and biplot.PCA
# ********************************************************

source("PCA.R")   # In the working directory or give path

# PCA; scaling 1 is the default for biplots
env.PCA.PL1 <- PCA(env, stand=TRUE)
biplot.PCA (env.PCA.PL1)
abline(h=0, lty=3)
abline(v=0, lty=3)
```

```
# PCA; scaling 2 in the biplot
env.PCA.PL2 <- PCA(env, stand=TRUE)
biplot.PCA (env.PCA.PL2 ,scaling=2)
abline(h=0, lty=3)
abline(v=0, lty=3)

 # The graphs may be mirror images of those obtained with vegan.
 # This is unimportant since the choice of the sign of the
 # principal components, made within the PCA functions, is
 # arbitrary.
```

5.4 Correspondence Analysis

5.4.1 Introduction

For a long time, CA has been one of the favourite tools for the analysis of species presence–absence or abundance data. The raw data are first transformed into a matrix $\overline{Q}$ of cell-by-cell contributions to the Pearson χ^2 statistic, and the resulting table is submitted to a singular value decomposition to compute its eigenvalues and eigenvectors. The result is an ordination, where it is the χ^2 distance (D_{16}) that is preserved among sites instead of the Euclidean distance D_1. The χ^2 distance is not influenced by the double zeros. Therefore, CA is a method adapted to the analysis of species abundance data without pre-transformation. Note that the data submitted to CA must be frequencies or frequency-like, dimensionally homogeneous and non-negative; that is the case of species counts or presence–absence data.

For technical reasons not developed here, CA ordination produces one axis fewer than min[*n,p*]. As in PCA, the orthogonal axes are ranked in decreasing order of the variation they represent, but instead of the total variance of the data, the variation is measured by a quantity called the total inertia (sum of squares of all values in matrix $\overline{Q}$, see Legendre and Legendre 1998, eq. 9.32). Individual eigenvalues are always smaller than 1. To know the amount of variation represented along an axis, one divides the eigenvalue of this axis by the total inertia of the species data matrix.

In CA, both the objects and the species are generally represented as points in the same joint plot. As in PCA, two scalings of the results are most useful in ecology. They are explained here for data matrices where objects are rows and species are columns:

- *CA scaling 1*: rows are at the centroids of columns. This scaling is the most appropriate if one is primarily interested in the ordination of *objects (sites)*.

In the multidimensional space, the χ^2 distance is preserved among objects. Interpretation: (1) the distances among objects in the reduced space approximate their χ^2 distances. Thus, object points that are close to one another are likely to be relatively similar in their species relative frequencies. (2) Any object found near the point representing a species is likely to contain a high contribution of that species. For presence–absence data, the object is more likely to possess the state "1" for that species.

- *CA scaling 2*: columns are at the centroids of rows. This scaling is the most appropriate if one is primarily interested in the ordination of *species*. In the multidimensional space, the χ^2 distance is preserved among species. Interpretation: (1) the distances among species in the reduced space approximate their χ^2 distances. Thus, species points that are close to one another are likely to have relatively similar relative frequencies along the objects. (2) Any species that lies close to the point representing an object is more likely to be found in that object, or to have a higher frequency there than in objects that are further away in the joint plot.

The Kaiser–Guttman criterion and the broken stick model, explained in Sect. 5.3.2.3, can be applied to CA axes for guidance as to the number of axes to retain. Our application below deals with the raw fish abundance data.

5.4.2 CA Using Function `cca()` of Package `vegan`

5.4.2.1 Running the Analysis and Drawing the Biplots

The procedure below closely resembles the one applied for PCA. First, let us run the analysis and draw the Kaiser–Guttman and broken stick plots (Fig. 5.5):

```
# CA of the raw species dataset (original species abundances)
# ***********************************************************

# Compute CA
spe.ca <- cca(spe)
spe.ca
summary(spe.ca)            # default scaling 2
summary(spe.ca, scaling=1)

 # The first axis has a large eigenvalue. In CA, values over 0.6
 # indicate a very strong gradient in the data. What proportion
 # of the total inertia does the first axis account for?
 # Note that the eigenvalues are the same in both scalings.
 # The scaling affects the eigenvectors but not the eigenvalues.
```

```
# Plot eigenvalues and % of variance for each axis
(ev2 <- spe.ca$CA$eig)
evplot(ev2)

 # Here the broken stick rule is more conservative than the
 # other.
 # The first axis is extremely dominant, as can be seen from
 # the bar plots as well as the numerical results.
```

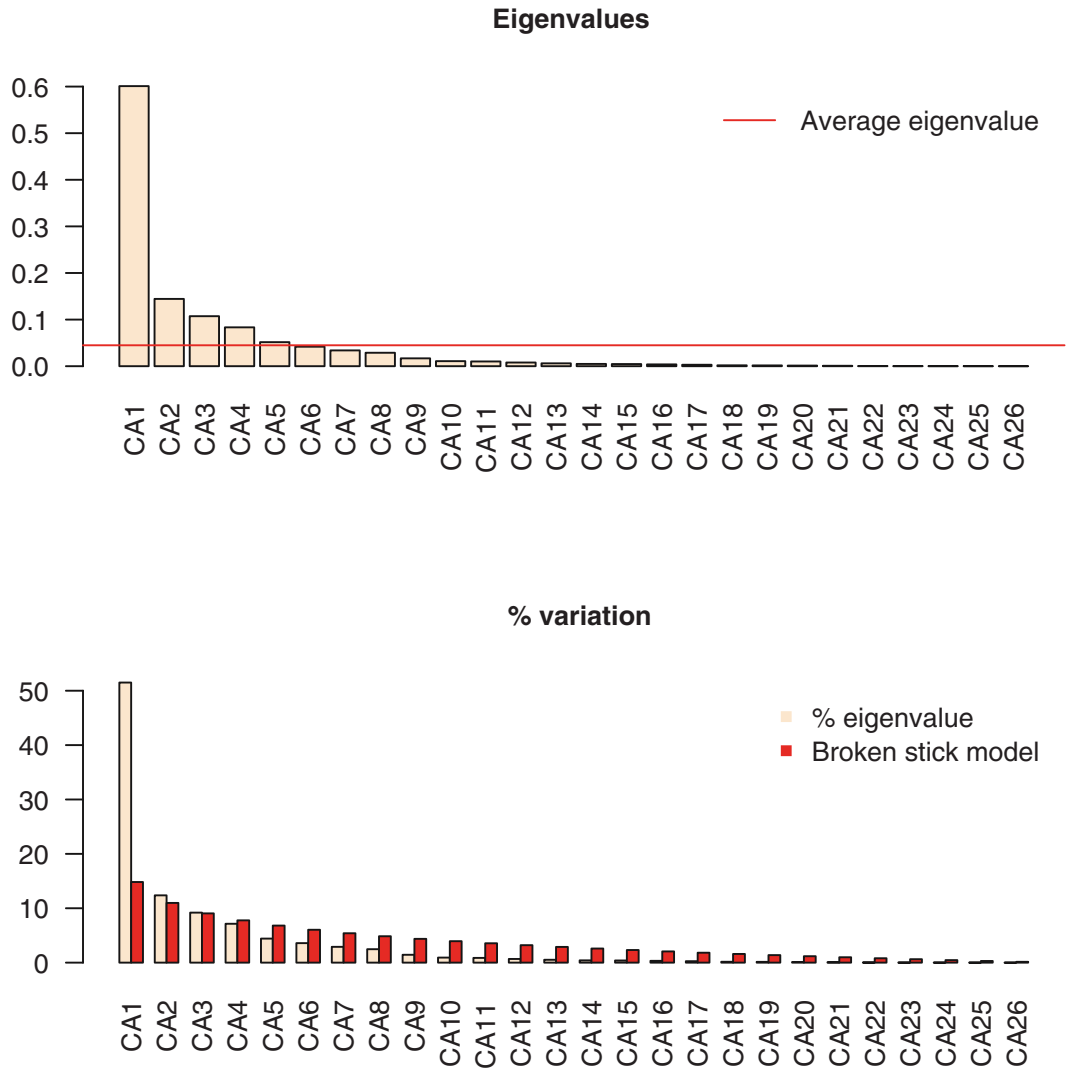

Fig. 5.5 Kaiser–Guttman and broken-stick plots to help assess the number of interpretable axes in CA. Application to the Doubs fish raw abundance data

It is time to draw the CA biplots of this analysis. Let us compare the two scalings (Fig. 5.6).

```
# CA biplots
# **********

par(mfrow=c(1,2))

# Scaling 1: sites are centroids of species
plot(spe.ca, scaling=1, main="CA fish abundances - biplot
  scaling 1")

# Scaling 2 (default): species are centroids of sites
plot(spe.ca, main="CA fish abundances - biplot scaling 2")
```

Hint *Here you could also produce a clustering and overlay its result on the CA plot.*

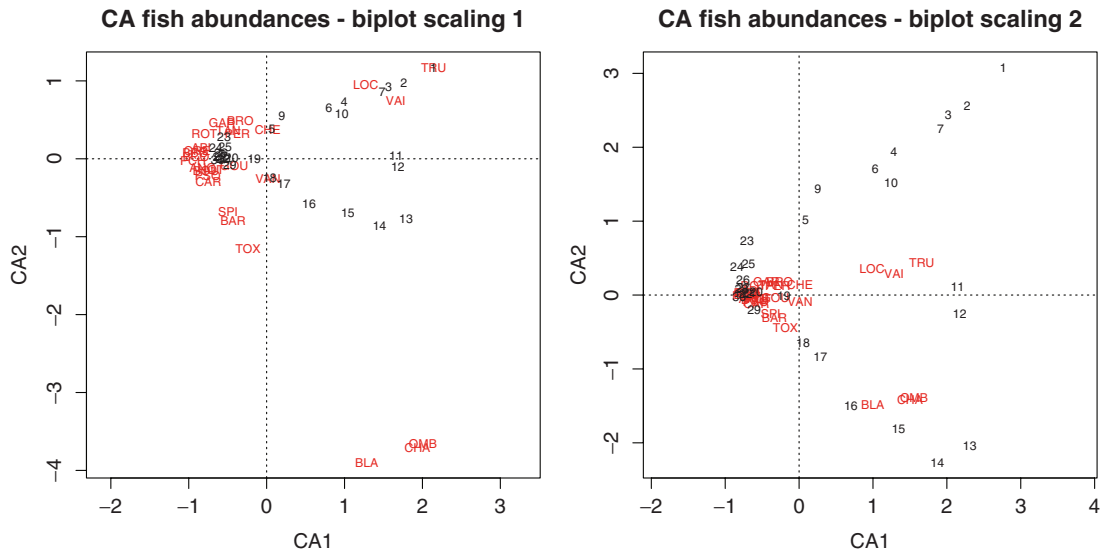

Fig. 5.6 CA biplots of the Doubs fish abundance data

The first axis opposes the lower section of the stream (sites 19–30) to the upper portion. This is clearly a strong contrast, which explains why the first eigenvalue is so high. Many species appear close to sites 19–30, indicating that they are more abundant downstream. Many of them are actually absent from the upper part of the river. The second axis contrasts the ten upstream sites to the intermediate ones. Both groups of sites, which display short gradients on their own, are associated with characteristic species. The scaling 2 plot shows how small groups of species are distributed among the sites. One can see that the grayling (OMB), the bullhead (CHA) and the varione (BLA) are found in the intermediate group of sites (11–18),

while the brown trout (TRU), the Eurasian minnow (VAI) and the stone loach (LOC) are found in a longer portion of the stream (approximately sites 1–18).

Observe how scalings 1 and 2 produce different plots. Scaling 1 shows the sites at the (weighted) centre of mass of the species. This is appropriate to interpret site proximities and find gradients or groups of sites. The converse is true for the scaling 2 biplot, where one can look for groups or replacement series of species. In both cases, care should be taken for the interpretation of species close to the origin of the graph. This proximity could mean either that the species is at its optimum in the mid-range of the ecological gradients represented by the axes, or that it is present everywhere along the gradient.

5.4.2.2 Passive (*Post Hoc*) Explanation of Axes Using Environmental Variables

Although there are means of incorporating explanatory variables directly in the ordination process (canonical ordination, see Chap. 6), one may be interested in interpreting a simple ordination by means of external variables. This can be done in **vegan** by means of the function **envfit()**. According to its author, Jari Oksanen, "**envfit** *finds vectors or factor averages of environmental variables. [...] The projections of points onto vectors have maximum correlation with corresponding environmental variables, and the factors show the averages of factor levels*".

The result is an object containing coordinates of factor levels (points) or arrowheads (quantitative variables) that can be used to project these variables into the ordination diagram (Fig. 5.7):

```
# A posteriori projection of environmental variables in a CA
# The last plot produced (CA scaling 2) must be active

spe.ca.env <- envfit(spe.ca, env)
plot(spe.ca.env)
# This has added the environmental variables to the last biplot
drawn

  # Does this new information help interpret the biplot?
```

Hint *This is a* post hoc *interpretation of ordination axes. Compare with Chap. 6.*

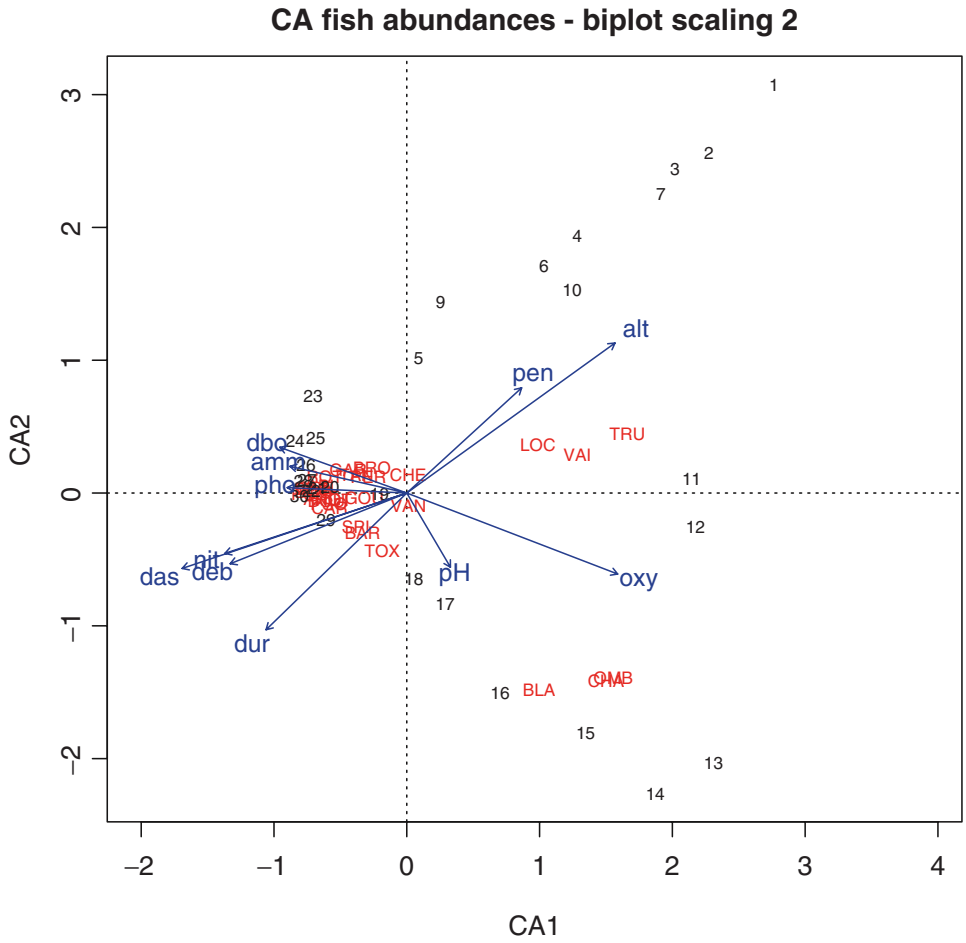

Fig. 5.7 CA biplot (scaling 2) of the Doubs fish abundance data with *a posteriori* projection of environmental variables

`envfit()` also proposes permutation tests to assess the significance of the r^2 of each explanatory variable regressed on the two axes of the biplot. But this is not, by far, the best way to test the effect of explanatory variables on a table of response variables. We explore this topic in Chap. 6.

5.4.2.3 Reordering the Data Table on the Basis of an Ordination Axis

A CA result is sometimes used to reorder the data table according to the first ordination axis. A compact form of ordered table is provided by the **vegan** function **vegemite()** already used in Chap. 4, which can use the information provided by an ordination computed in **vegan**:

```
# Species data table ordered after the CA result
# *********************************************

vegemite(spe, spe.ca)

  # The left-right and up-down orderings in this ordered
  # table depends on the (arbitrary) orientation of the
  # ordination axes.
  # Observe that the ordering is not optimal since it is done
  # only on the basis of the first axis. Therefore, sites
  # 1 to 10 and 11 to 18 (separated along axis 2) and their
  # corresponding characteristic species are interspersed.
```

5.4.3 CA Using Function CA()

As in the case of PCA, we propose a simple CA function: **CA()**. Here is how to use it on the fish data.

```
# CA using CA() function
# **********************

source("CA.R") # Function in the working directory or give path
spe.CA.PL <- CA(spe)
biplot(spe.CA.PL, cex=1)

# Ordering of the data table following the first CA axis
# The table is transposed, as in vegemite() output
summary(spe.CA.PL)
t(spe[order(spe.CA.PL$F[,1]),order(spe.CA.PL$V[,1])])
```

Hints *The use of matrices F and V to reorder the table relates to the symbolism used by Legendre and Legendre (1998, Section 9.4) to explain the mathematics of correspondence analysis. Using Vhat instead of F and Fhat instead of V (i.e. using the scaling 2 projection) would have produced the same ordered table.*

Argument `cex` *of the* **biplot()** *function is here to adapt the size of the symbols and the site and species names to the plot. The default is* `cex=2`*. Smaller values produce smaller symbols and characters. They may be useful for plots containing many sites and species.*

5.4.4 Arch Effect and Detrended Correspondence Analysis

Long environmental gradients often support a succession of species. Since the species that are controlled by environmental factors tend to have unimodal distributions, a long gradient may encompass sites that, at both ends of the gradient, have no species in common; thus, their distance reaches a maximum value (or their similarity is 0). But if one looks at either side of the succession, contiguous sites continue to grow more different from each other. Therefore, instead of a linear trend, the gradient is represented on a pair of CA axes as an arch. Several detrending techniques have been proposed to counter this effect and straighten up gradients in ordination diagrams, leading to detrended correspondence analysis (DCA):

- Detrending by segments combined with nonlinear rescaling: axis I is divided into an arbitrary number of segments and, within each one, the mean of the object scores along axis 2 is made equal to zero. The number of segments has a large influence on the result. The DCA results presented in the literature suggest that the scores along the second axis are essentially meaningless. The authors of this book strongly warn against the use of this form of DCA as an ordination technique; however, it may be used to estimate the "gradient length" of the first ordination axis, expressed in standard deviation units of species turnover. A gradient length larger than 4 indicates that some species have a unimodal response along the axis (ter Braak and Šmilauer 2002).
- Detrending by polynomials: another line of reasoning about the origin of the arch effect leads to the observation that when an arch occurs, the second axis can be seen as quadratically related to the first (i.e. it is the first axis to the power 2). This explains for the parabolic shape of the scatter of points. Hence, a solution is to make the second axis not only linearly, but also quadratically independent from the first. Although intuitively attractive, this method of detrending has to be applied with caution because it actually imposes a constraining model on the data.

DCA by segments is available in package **`vegan`** (function **`decorana()`**). In the output of this function, the gradient length of the axes is called "Axis lengths".

Given all its problems (see discussion in Legendre and Legendre 1998, pp. 465–472), we do not describe this method further here.

An even more extreme effect of the same kind exists in PCA. It is called the horseshoe effect because, in the case of strong gradients, the sites of both ends bend inwards and appear closer than other pairs. This is due to the fact that PCA considers double zeros as resemblances. Consequently, sites located at opposite ends of an ecological gradient, having many double zeros, "resemble" each other on this respect. The Hellinger or chord transformation of the species data partly alleviates this problem.

5.4.5 Multiple Correspondence Analysis

Multiple correspondence analysis (MCA) is the counterpart of PCA for the ordination of a table of categorical variables, i.e. a data frame in which all variables are factors. It is implemented in the function **`mca()`** of the package **`MASS`** and, with more options, in the function **`MCA()`** of the package **`FactoMineR`**.

5.5 Principal Coordinate Analysis

5.5.1 Introduction

PCA as well as CA impose the distance preserved among objects: the Euclidean distance (and several others with pre-transformations) for PCA and the χ^2 distance for CA. If one wishes to ordinate objects on the basis of another distance measure, more appropriate to the problem at hand, then PCoA is the method of choice. It provides a Euclidean representation of a set of objects whose relationships are measured by any similarity or distance measure chosen by the user. For example, if the coefficient is Gower's index S_{15}, which can combine descriptors of many mathematical types into a single measure of resemblance, then the ordination represents the relationships among the objects based upon these many different variables. This would not be possible with PCA or CA.

Like PCA and CA, PCoA produces a set of orthogonal axes whose importance is measured by eigenvalues. Since it is based on an association matrix, it can directly represent the relationships either among objects (if the association matrix was in Q mode) or variables (if the association matrix was in R mode). If it is necessary to project variables, e.g. species, on a PCoA ordination of the objects, the variables can be related *a posteriori* to the ordination axes using correlations or weighted averages and drawn on the ordination plot. In the case of Euclidean association measures, PCoA behaves in a Euclidean manner. For instance, computing a Euclidean distance among sites and running a PCoA yields the same results as running a PCA on a covariance matrix of the same data and looking at the scaling 1 ordination results. But if the association coefficient used is non-Euclidean, then

PCoA may react by producing several negative eigenvalues in addition to the positive ones (and a null eigenvalue in-between). The axes corresponding to negative eigenvalues cannot be represented on real ordination axes since they are complex. In most applications, this does not affect the representation of the objects on the several first principal axes, but it can lead to problems if the largest negative eigenvalues are of the same magnitude in absolute value as the first positive ones.

There are technical solutions to this problem, which consist in adding a constant to either the squared distances among objects (Lingoes correction) or to the distances themselves (Cailliez correction) (Gower and Legendre 1986). In the function **`cmdscale()`** presented below, the Cailliez correction is obtained with the argument `add=TRUE`.

One can avoid complex axes by keeping the eigenvectors with their original Euclidean norm (vector length = 1) instead of dividing each one by the square root of its eigenvalue, as is usual in the PCoA procedure. This workaround is used in the MEM spatial analysis presented in Chap. 7. It should not be used for routine ordination by PCoA since eigenvectors that have not been rescaled to $\sqrt{\text{eigenvalue}}$ cannot be used to produce plots that preserve the original distances among the objects.

The ordination axes of a PCoA can be interpreted like those of a PCA or CA: proximity of objects represents similarity in the sense of the association measure used.

5.5.2 *Application to the Doubs Data Set Using* ***`cmdscale`*** *and* ***`vegan`***

As an example, let us compute a matrix of Bray–Curtis dissimilarities among sites, and subject this matrix to PCoA. In **`vegan`**, there is a way to project weighted averages of species abundances on a PCoA plot, by means of function **`wascores()`** (Fig. 5.8). Since species are projected as weighted averages of their contributions to the sites, their interpretation with respect to the sites is done as in CA.

```
# PCoA on a Bray-Curtis dissimilarity matrix of fish species
# ***********************************************************
spe.bray <- vegdist(spe)
spe.b.pcoa <- cmdscale(spe.bray, k=(nrow(spe)-1), eig=TRUE)

# Plot of the sites and weighted average projection of species
ordiplot(scores(spe.b.pcoa)[,c(1,2)], type="t",
 main="PCoA with species")
abline(h=0, lty=3)
abline(v=0, lty=3)
```

```
# Add species
spe.wa <- wascores(spe.b.pcoa$points[,1:2], spe)
text(spe.wa, rownames(spe.wa), cex=0.7, col="red")
```

Hint *Observe the use of two* **vegan** *functions,* **ordiplot()** *and* **scores()**, *to produce the ordination plot.* **vegan** *is a world in itself and often requires special functions to handle its own results.*

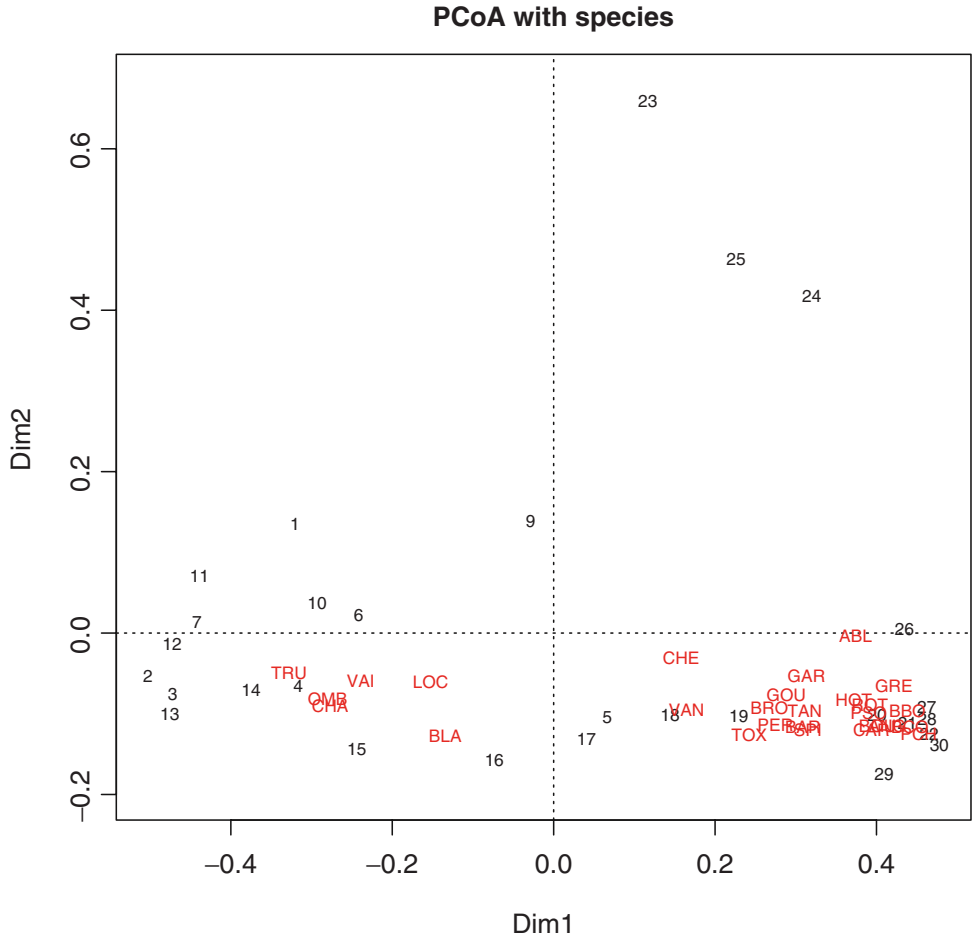

Fig. 5.8 PCoA biplot of a Bray–Curtis dissimilarity matrix of the raw Doubs fish abundance data. *A posteriori* projection of the species as weighted averages. The relationships between species and sites are interpreted as in CA

5.5.3 Application to the Doubs Data Set Using `pcoa()`

There is another way to achieve double projection. It is based on correlations of the environmental variables with the PCoA ordination axes (see Legendre and Legendre 1998, p. 431). If a PCoA of a matrix of Euclidean distances and scaling 1 is computed, this method produces vectors corresponding to what would be obtained in a scaling 1 PCA biplot of the same data. This representation is available in functions **`pcoa()`** and **`biplot.pcoa()`**, both available in packages **`ape`** and **`PCNM`**.

Here is how these functions work. In our example, PCoA is run on a Euclidean distance matrix computed on a Hellinger-transformed species abundance matrix; the result of these two operations is a Hellinger distance matrix. In such a case, it is actually better (simpler and faster) to run a PCA directly on the transformed species data, but here the idea is to allow a comparison with the PCA run presented in Sect. 5.3.3. Two biplots are proposed, with projection of the raw and standardized species abundances. Compare the result below (Fig. 5.9) with the biplot of the PCA scaling 1 result.

```
# PCoA and projection of species vectors using function pcoa()
# ************************************************************

spe.h.pcoa <- pcoa(dist(spe.h))

# Biplots
par(mfrow=c(1,2))
# First biplot: Hellinger-transformed species data
biplot.pcoa(spe.h.pcoa, spe.h, dir.axis2=-1)
abline(h=0, lty=3)
abline(v=0, lty=3)
# Second biplot: standardized Hellinger-transformed species data
spe.std <- apply(spe.h, 2, scale)
biplot.pcoa(spe.h.pcoa, spe.std, dir.axis2=-1)
abline(h=0, lty=3)
abline(v=0, lty=3)

 # How does this result compare with that of the PCA?
```

Hints *For projection of species data onto a PCoA plot, it is important to use the species data with the same transformation (if any) as the one used to compute the dissimilarity matrix. The standardization proposed here as an alternative may help better visualize the variables if they have very different variances.*

The argument `dir.axis2=-1` *reverses axis 2 to make the result directly comparable with the PCA result in Fig. 5.4, scaling 1. Remember that the signs of ordination axes are arbitrary.*

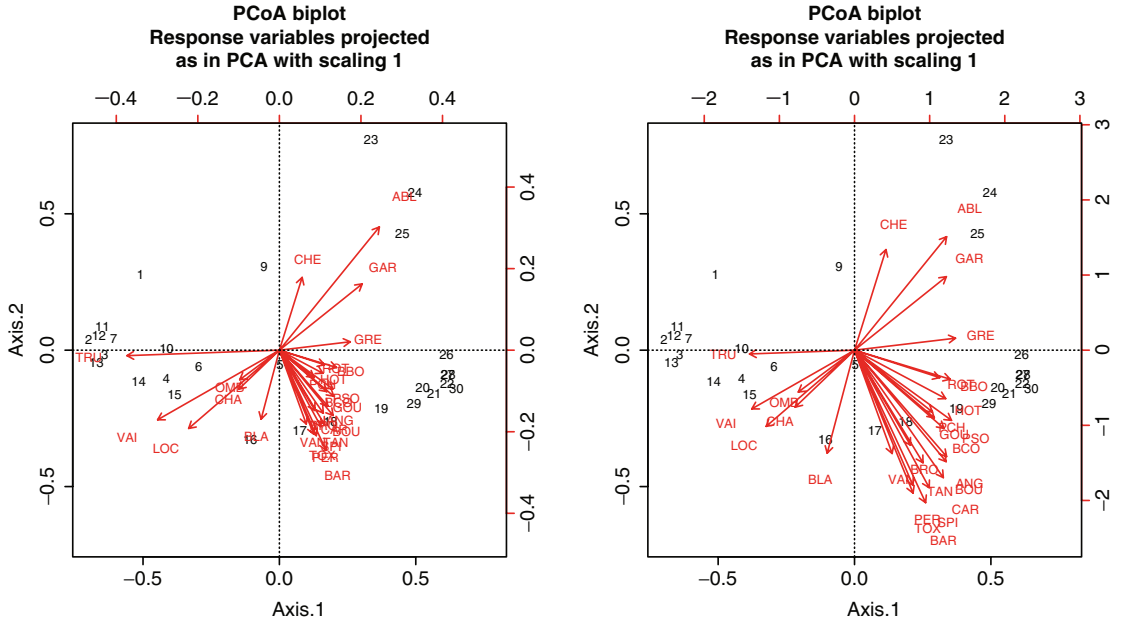

Fig. 5.9 PCoA biplots of the fish data obtained with functions **`pcoa()`** and **`biplot.pcoa()`**. *Left*: Hellinger-transformed raw species variables. *Right*: standardized Hellinger-transformed species. The bottom and left-hand scales are for the objects, the top and right-hand scales are for the species

As mentioned above, PCoA should actually be reserved to situations where no Euclidean measure is available or selected. With Jaccard and Sørensen dissimilarity matrices computed by **`ade4`**, for example, the ordination is fully Euclidean. In other cases, however, such as Bray–Curtis dissimilarities computed with **`vegan`**, the dissimilarity matrices may not be Euclidean (see Sect. 3.3.5). This results in PCoA producing some negative eigenvalues. Lingoes and Cailliez corrections are available in the function **`pcoa()`**. This function provides the eigenvalues along with a broken stick comparison in its output. Compare the examples below:

```
# Comparison of PCoA results with Euclidean and non-Euclidean
# dissimilarity matrices
# ********************************************************

# PCoA on a Hellinger distance matrix
is.euclid(dist(spe.h))
summary(spe.h.pcoa)
spe.h.pcoa$values
```

```
# PCoA on a Bray-Curtis dissimilarity matrix
is.euclid(spe.bray)
spe.bray.pcoa <- pcoa(spe.bray)
spe.bray.pcoa$values     # Observe eigenvalues 18 and following

# PCoA on the square root of a Bray-Curtis dissimilarity matrix
is.euclid(sqrt(spe.bray))
spe.braysq.pcoa <- pcoa(sqrt(spe.bray))
spe.braysq.pcoa$values  # Observe the eigenvalues

# PCoA on a Bray-Curtis dissimilarity matrix with Lingoes
correction
spe.brayl.pcoa <- pcoa(spe.bray, correction = "lingoes")
spe.brayl.pcoa$values   # Observe the eigenvalues

# PCoA on a Bray-Curtis dissimilarity matrix with Cailliez
correction
spe.brayc.pcoa <- pcoa(spe.bray, correction = "cailliez")
spe.brayc.pcoa$values   # Observe the eigenvalues

 # If you want to choose the analysis displaying the highest
 # proportion of variation on axes 1+2, which solution will you
 # select among those above?
```

5.6 Nonmetric Multidimensional Scaling

5.6.1 Introduction

If the researcher's priority is not to preserve the exact distances among objects in an ordination plot, but rather to represent as well as possible the ordering relationships among objects in a small and specified number of axes, nonmetric multidimensional scaling (NMDS) may be the solution. Like PCoA, NMDS can produce ordinations of objects from any distance matrix. The method can cope with missing distances, as long as there are enough measures left to position each object with respect to a few others. NMDS is not an eigenvalue technique, and it does not maximize the variability associated with individual axes of the ordination. As a result, plots may arbitrarily be rotated, centred or inverted. The procedure goes as follows (very schematically; for details see Legendre and Legendre 1998, p. 445 *et seq.*):

- Specify the number *m* of axes (dimensions) sought.
- Construct an initial configuration of the objects in the *m* dimensions, to be used as a starting point of an iterative adjustment process. This is a tricky step, since

the end-result may depend on the starting configuration. A PCoA ordination may be a good starting point. Otherwise, try many independent runs with random initial configurations.
- An iterative procedure tries to position the objects in the requested number of dimensions in such a way as to minimize a stress function (scaled from 0 to 1), which measures how far the distances in the reduced-space configuration are from being monotonic to the original distances in the association matrix.
- The adjustment goes on until the stress value can no more be lowered, or until it reaches a predetermined value (tolerated lack-of-fit).
- Most NMDS programs rotate the final solution using PCA for easier interpretation.

For a given and small number of axes (e.g. $m = 2$ or 3), NMDS often achieves a less deformed representation of the distance relationships among objects than a PCoA in the same number of dimensions. But NMDS is a computer-intensive technique exposed to the risk of suboptimal solutions in the iterative process. Indeed, the objective stress function to minimize often reaches a local minimum larger than the true minimum.

5.6.2 Application to the Fish Data

NMDS can be performed in **R** with the elegant function **`metaMDS()`** from the **`vegan`** package. **`metaMDS()`** accepts raw data or distance matrices. Let us apply it to the fish abundances using the Bray–Curtis index. **`metaMDS()`** uses random starts and iteratively tries to find the best possible solution. Species points are added to the ordination plot using **`wascores()`**. See Fig. 5.10.

If one must use a distance matrix with missing values, NMDS can be computed with the function **`isoMDS()`**. An initial configuration must be provided in the form of a matrix positioning the sites (argument `y`) in the number of dimensions specified for the analysis (argument `k`). To reduce the risk of reaching a local minimum, we suggest to use the function **`bestnmds()`** of the package **`labdsv`**. This function, which is a wrapper for **`isoMDS()`**, computes the analysis a user-specified number of times (argument `itr`) with internally produced random initial configurations. The solution with the smallest stress value is retained by the function.

```
# NMDS applied to the fish species - Bray-Curtis distance matrix
# ************************************************************

spe.nmds <- metaMDS(spe, distance="bray")
spe.nmds
```

```
spe.nmds$stress
plot(spe.nmds, type="t", main=paste("NMDS/Bray - Stress =",
 round(spe.nmds$stress,3)))

 # How does this result compare with those of PCA, CA and PCoA?
```

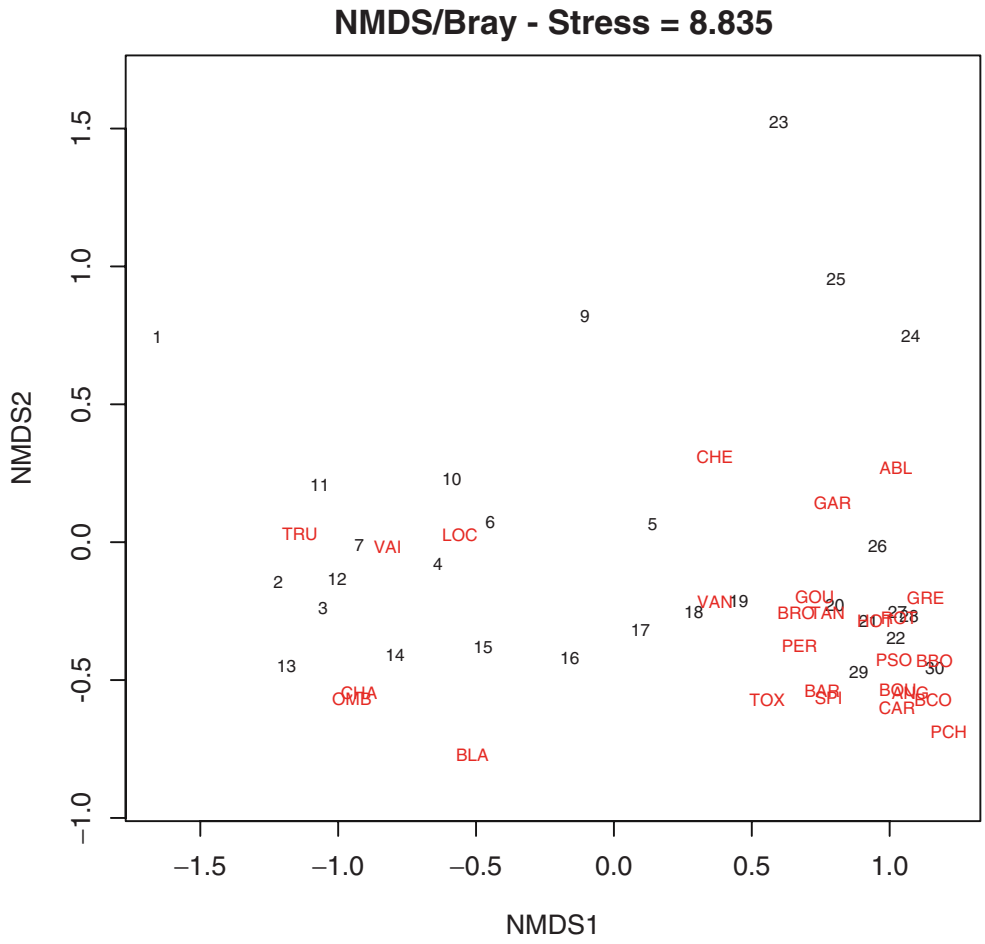

Fig. 5.10 NMDS biplot of a Bray–Curtis dissimilarity matrix of the fish abundance data. Species added using weighted averages. The relationships between species and sites are interpreted as in CA

A useful way to assess the appropriateness of an NMDS result is to compare, in a *Shepard diagram,* the distances among objects in the ordination plot with the original distances. In addition, the goodness-of-fit of the ordination is measured as the R^2

of either a linear or a non-linear regression of the NMDS distances on the original ones. All this is possible in **R** using **vegan**'s functions **stressplot()** and **goodness()** (Fig. 5.11):

```
# Shepard plot and goodness of fit
# ********************************

par(mfrow=c(1,2))
stressplot(spe.nmds, main="Shepard plot")
gof = goodness(spe.nmds)
plot(spe.nmds, type="t", main="Goodness of fit")
points(spe.nmds, display="sites", cex=gof*2)
```

Hint *See how the goodness-of-fit of individual sites is represented using the results of the* **goodness()** *analysis by way of the* `cex` *argument of the* **points()** *function. Poorly fitted sites have larger bubbles.*

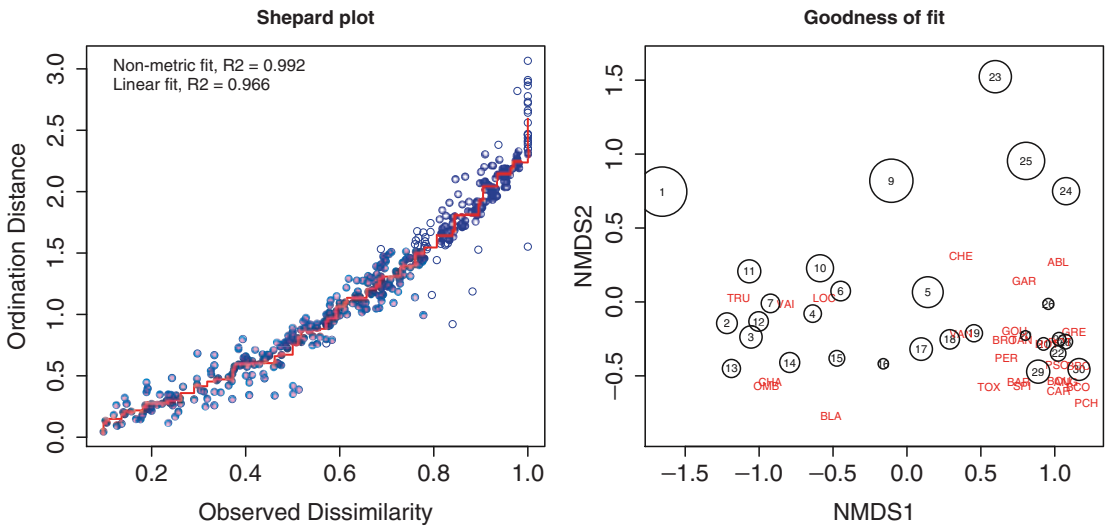

Fig. 5.11 Shepard and goodness-of-fit diagrams of the NMDS result presented in Fig. 5.10

As with the other ordination methods, it is possible to add information coming from a clustering result to an NMDS ordination plot. For instance, compute a Ward clustering of the Bray–Curtis matrix, extract four groups and colourize the sites according to them:

```
# Add colours from a clustering result to an NMDS plot
# ***************************************************

# Ward clustering of Bray-Curtis dissimilarity matrix
# and extraction of four groups
spe.bray.ward <- hclust(spe.bray, "ward")
spe.bw.groups <- cutree(spe.bray.ward, k=4)
grp.lev <- levels(factor(spe.bw.groups))

# Combination with NMDS result
sit.sc <- scores(spe.nmds)
p <- ordiplot(sit.sc, type="n",
 main="NMDS/Bray + clusters Ward/Bray")
for (i in 1:length(grp.lev)) {
 points(sit.sc[spe.bw.groups==i,], pch=(14+i), cex=2, col=i+1)
 }
text(sit.sc, row.names(spe), pos=4, cex=0.7)
# Add the dendrogram
ordicluster(p, spe.bray.ward, col="dark grey")
legend(locator(1), paste("Group",c(1:length(grp.lev))),
 pch=14+c(1:length(grp.lev)), col=1+c(1:length(grp.lev)),
 pt.cex=2)
```

5.7 Handwritten Ordination Function

To conclude this chapter, let us dive into the bowels of an ordination method…

The Code It Yourself corner #3

Legendre and Legendre (1998) provide the algebra necessary to program the ordination methods seen above directly "from scratch", i.e., using the matrix algebra functions implemented in ***R****. While it is not the purpose of this book to do this for all methods, we provide an example that could stimulate the interest among users. After all, numerical ecology is a living science, and anybody could one day stumble upon a situation for which no ready-made function exists. The researcher may then be interested in developing his or her own method and write the functions to implement it.*

The example below is based on the algebraic development presented in Legendre and Legendre (1998), Section 9.1. It is presented in the form of a function, the kind that any user could write for her or his own use. The steps are the following for a PCA on a covariance matrix. To obtain a PCA on a correlation matrix, one has to standardize the data before using the function, or implement this as an option in the function itself.

1. *Compute the covariance matrix S of the original or centred data matrix.*
2. *Compute the eigenvectors and eigenvalues of S (eqs. 9.1 and 9.2).*

3. *Extract matrix U of the eigenvectors and compute matrix F of the principal components (eq. 9.4) for the scaling 1 biplot.*
4. *Computation of matrices U2 and G for the scaling 2 biplot.*
5. *Output of the results.*

```
# A simple function to perform PCA

   myPCA <- function(Y) {
   Y.mat <- as.matrix(Y)
   object.names <- rownames(Y)
   var.names <- colnames(Y)

# Centre the data (will be needed to compute F)
   Y.cent <- scale(Y.mat, center=TRUE, scale=FALSE)

# Covariance matrix S
   Y.cov <- cov(Y.cent)

# Eigenvectors and eigenvalues of S (eq. 9.1 and 9.2)
   Y.eig <- eigen(Y.cov)

# Copy the eigenvectors to matrix U (used to represent
# variables in scaling 1 biplots)
   U <- Y.eig$vectors
   rownames(U) <- var.names

# Compute matrix F (used to represent objects
# in scaling 1 plots)
   F <- Y.cent%*%U       # eq. 9.4
   rownames(F) <- object.names

# Compute matrix U2 (to represent variables in scaling 2
# plots) Legendre and Legendre 1998, unnumbered equation p. 397)
   U2 <- U%*%diag(Y.eig$values^0.5)
   rownames(U2) <- var.names

# Compute matrix G (to represent objects in scaling 2 plots)
# Legendre and Legendre 1998, unnumbered equation p. 404)
   G <- F%*%diag(Y.eig$values^0.5)
   rownames(G) <- object.names

# Output of a list containing all the results
   result <- list(Y.eig$values,U,F,U2,G)
   names(result) <- c("eigenvalues","U", "F", "U2", "G")
   result
   }
```

This function should give exactly the same results as the function **PCA()** *used in Section 5.3.5. Now try it on the Hellinger-transformed fish species data and compare the results.*

To make your function active, either **save it in a file** *(called for instance* **myPCA.R***) and source it, or (less elegant) copy the whole code directly into your* ***R*** *console.*

```
# PCA on fish species using hand-written function
fish.PCA <- myPCA(spe.h)
summary(fish.PCA)
# Eigenvalues
fish.PCA$eigenvalues
# Eigenvalues expressed as percentages
round(100*fish.PCA$eigenvalues/sum(fish.PCA$eigenvalues),2)
# Alternate computation of total variation (denominator)
round(100*fish.PCA$eigenvalues/sum(diag(cov(spe.h))),2)
# Cumulative eigenvalues expressed as percentages
round(cumsum(100*fish.PCA$eigenvalues/sum(fish.PCA$eigenvalues))
,2)

# Biplots
par(mfrow=c(1,2))
# Scaling 1 biplot
biplot(fish.PCA$F, fish.PCA$U)
# Scaling 2 biplot
biplot(fish.PCA$G, fish.PCA$U2)
```

Now you could plot other pairs of axes, for instance axes 1 and 3. Compared to CA or PCoA, the code above is rather straightforward. But nothing prevents you from trying to program another method. You can also display the code of the **CA()** *and* **pcoa()** *functions and interpret them with the manual in hand.*

Chapter 6
Canonical Ordination

6.1 Objectives

Simple (unconstrained) ordination analyses one data matrix and reveals its major structure in a graph constructed from a reduced set of orthogonal axes. It is therefore a passive form of analysis, and the user interprets the ordination results *a posteriori*, as described in Chap. 5. Canonical ordination, on the contrary, associates two or more data sets in the ordination process itself. Consequently, if one wishes to extract structures of a data set that are related to structures in other data sets, and/or formally test statistical hypotheses about the significance of these relationships, canonical ordination is the way to go.

Canonical ordination methods can be classified into two groups: symmetrical and asymmetrical.

Practically, you will:

- Learn how to choose among various canonical ordination techniques: redundancy analysis (RDA), distance-based redundancy analysis (db-RDA), canonical correspondence analysis (CCA), linear discriminant analysis (LDA), canonical correlation analysis (CCorA), co-inertia analysis (CoIA) and multiple factor analysis (MFA)
- Compute them using the correct options and properly interpret the results
- Apply these techniques to the Doubs River data
- Explore particular applications of some canonical ordination methods, for instance, variation partitioning and multivariate analysis of variance (MANOVA) by RDA
- Write your own RDA function

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_6, © Springer Science+Business Media, LLC 2011

6.2 Canonical Ordination Overview

In the methods explored in Chap. 5, the ordination procedure itself is not influenced by external variables; these may be only considered after the computation of the ordination. One lets the data matrix express the relationships among objects and variables without constraint. This is an exploratory, descriptive approach. Canonical ordination, on the contrary, explicitly explores the relationships between two matrices: a response matrix and an explanatory matrix in some cases (asymmetrical analysis), and two matrices with symmetrical roles in other cases. Both matrices are used in the production of the ordination.

The way to combine the information of two (or, in some cases, more) data matrices depends on the method of analysis. We first explore the two asymmetrical methods that are mostly used in ecology nowadays, i.e. redundancy analysis (RDA) and canonical correspondence analysis (CCA). Both combine multiple regression with classical ordination (PCA or CA). Partial RDA is also explored, as well as a procedure of variation partitioning based on this method. The significance of canonical ordinations are tested by means of permutations. After that, we devote short sections to linear discriminant analysis (LDA), which looks for a combination of quantitative variables to explain a predefined grouping of the objects, and to canonical correlation analysis (CCorA), co-inertia analysis (CoIA) and multiple factor analysis (MFA), three symmetrical methods that compute eigenvectors describing the common structure of two or several data sets.

6.3 Redundancy Analysis

6.3.1 Introduction

RDA is a method combining regression and principal component analysis (PCA). It is a direct extension of regression analysis to model multivariate response data. RDA is an extremely powerful tool in the hands of ecologists, especially since the introduction of the Legendre and Gallagher (2001) transformations that open RDA to the analysis of community composition data tables (transformation-based RDA or tb-RDA).

Conceptually, RDA is a multivariate (meaning *multiresponse*) multiple linear regression followed by a PCA of the table of fitted values. It works as follows, on a matrix **Y** of **centred** response data and a matrix **X** of centred (or, more generally, standardized) explanatory variables:

- Regress each (centred) **y** variable on explanatory table **X** and compute the fitted ($\hat{y}$) (this is the only required matrix in most analyses) and residual (y_{res}) vectors (if needed). Assemble all vectors $\hat{\mathbf{y}}$ into a matrix $\hat{\mathbf{Y}}$ of fitted values
- Compute a PCA of the matrix $\hat{\mathbf{Y}}$ of fitted values; this analysis produces a vector of canonical eigenvalues and a matrix **U** of *canonical eigenvectors*
- Use matrix **U** to compute *two types of ordination site scores*: use either the original centred matrix **Y** to obtain an ordination in the space of the original variables **Y** (i.e. compute **YU**, obtaining site scores called "*Site scores (weighted sums of site scores)*" in **vegan**), or use the matrix $\hat{\mathbf{Y}}$ of fitted values to obtain an ordination in the space of variables **X** (i.e. compute $\hat{\mathbf{Y}}$**U**, which produces fitted site scores called "*Site constraints (linear combinations of constraining variables)*" in **vegan**)
- The residual values from the multiple regressions (i.e. $\mathbf{Y}_{res} = \mathbf{Y} - \hat{\mathbf{Y}}$) may also be submitted to a PCA to obtain an unconstrained ordination of the residuals. This partial PCA, which is not strictly speaking part of the RDA, is computed by **vegan**'s `rda()` function

Additional information on the algebra of RDA is presented in the **Code it yourself** corner at the end of this section.

As can be seen from the conceptual steps presented above, RDA computes axes that are linear combinations of the explanatory variables. In other words, this method seeks, in successive order, a series of linear combinations of the explanatory variables that best explain the variation of the response matrix. The axes defined in the space of the explanatory variables are orthogonal to one another. RDA is therefore a constrained ordination procedure. The difference with unconstrained ordination is important: the matrix of explanatory variables conditions the "weights" (eigenvalues), the orthogonality and the direction of the ordination axes. In RDA, one can truly say that the axes *explain* or *model* (in the statistical sense) the variation of the dependent matrix. Furthermore, a hypothesis (H_0) of absence of linear relationship between **Y** and **X** can be tested in RDA; this is not the case in PCA.

An RDA produces min[p, m, $n-1$] canonical axes, where n is the number of objects and m is the number of degrees of freedom of the model (number of numeric explanatory variables, including levels of factors if qualitative explanatory variables are included; a factor with k classes requires ($k-1$) dummy variables for coding, so there are ($k-1$) degrees of freedom for this factor). Each of the canonical axes is a linear combination (i.e. a multiple regression model) of *all* explanatory variables. RDA is usually computed, for convenience, on standardized explanatory variables; the fitted values of the regressions, as well as the canonical analysis results, are unchanged

by standardization of the **X** variables.[1] In **vegan**'s **rda()** function, the variation of the data matrix that cannot be explained by the environmental variables (i.e. the residuals of the regressions) is expressed by the unconstrained PCA eigenvalues, which are given after the canonical eigenvalues.

For the reasons explained in Chap. 5 about PCA, an RDA can be computed on a covariance or a correlation response matrix. To obtain an analysis on the correlation response matrix, standardization of the response data is done by the option `scale=TRUE` in **vegan**'s **rda()**.

The statistical significance of an RDA (global model) and that of individual canonical axes can be tested by permutations. These tests are introduced in due course.

6.3.2 RDA of the Doubs River Data

You will now explore various aspects of RDA. To achieve this, you will first prepare not only the data sets as usual, but also transform a variable and divide the explanatory variables into two subsets.

6.3.2.1 Preparation of the Data

```
# Load required  packages
library(ade4)
library(vegan)
library(packfor) # Available on R-Forge,
          # URL= https://r-forge.r-project.org/R/?group_id=195
# In MacOS X, the gfortran package is required by the
# forward.sel() function of packfor, so users must install the
# gfortran compiler. Choose 'MacOS X' in the
# 'cran.r-project.org' window, then 'tools'.
library(MASS)
library(ellipse)
library(FactoMineR)

# Additional functions
source("evplot.R")
source("hcoplot.R")
```

[1] Note also that in many cases the explanatory variables are not dimensionally homogeneous. In such cases an effect of standardization is that the absolute values of the canonical coefficients (i.e. the regression coefficients of the models) measure which variable(s) is or are most important to explain each canonical axis.

```
# Import the data from CSV files
spe <- read.csv("DoubsSpe.csv", row.names=1)
env <- read.csv("DoubsEnv.csv", row.names=1)
spa <- read.csv("DoubsSpa.csv", row.names=1)
# Remove empty site 8
spe <- spe[-8,]
env <- env[-8,]
spa <- spa[-8,]

# Set aside the variable 'das' (distance from the source) for
later use
das <- env[,1]

# Remove the 'das' variable from the env dataset
env <- env[,-1]
# Recode the slope variable (pen) into a factor (qualitative)
# variable (to show how these are handled in the ordinations)
pen2 <- rep("very_steep", nrow(env))
pen2[env$pen <= quantile(env$pen)[4]] = "steep"
pen2[env$pen <= quantile(env$pen)[3]] = "moderate"
pen2[env$pen <= quantile(env$pen)[2]] = "low"
pen2 <- factor(pen2, levels = c("low", "moderate", "steep",
  "very_steep"))
table(pen2)
# Create an env2 data frame with slope as a qualitative variable
env2 <- env
env2$pen <- pen2

# Create two subsets of explanatory variables
# Physiography (upstream-downstream gradient)
envtopo <- env[,c(1:3)]
names(envtopo)
# Water quality
envchem <- env[,c(4:10)]
names(envchem)

# Hellinger-transform the species dataset
spe.hel <- decostand(spe, "hellinger")
```

6.3.2.2 RDA Using **vegan**

vegan allows the computation of an RDA in two different ways. The simplest syntax is to list the names of the objects involved separated by commas:

```
simpleRDA <- rda(Y,X,W)
```

where **Y** is the response matrix, **X** is the matrix of explanatory variables and **W** is an optional matrix of covariables (variables whose variation is to be controlled for in a partial analysis).

This call, although simple, has some limitations. Its main drawback is that it does not allow factors (qualitative variables) to be included in the explanatory and covariable matrices. Therefore, in all but the simplest applications, it is better to use the formula interface:

```
formulaRDA <- rda(Y ~ var1 + factorA + var2*var3 +
                      Condition(var4), data=XWdata)
```

In this example, **Y** is the response matrix; the constraint includes a quantitative variable (`var1`), a factor (`factorA`), an interaction term between variables 2 and 3, whereas the effect of `var4` is partialled out. The explanatory variables and the covariable are in object `XWdata`, which must have the class data.frame.

This is the same kind of formula as used in **`lm()`** and other **R** functions devoted to regression. We use it in the example below. For more information about this topic, consult the **`rda()`** help file.

```
# RDA of the Hellinger-transformed fish species data constrained
# by all the environmental variables contained in env2
# Observe the shortcut formula

spe.rda <- rda(spe.hel ~ ., env2)
summary(spe.rda) # Scaling 2 (default)

 # Here the default choices are used, i.e. scale=FALSE (RDA on
 # a covariance matrix) and scaling=2.
```

Hint *See the shortcut to use all variables present in object* `env2`*, without having to name them.*

Here is an excerpt of the output:

```
 Call:
rda(formula = spe.hel ~ alt + pen + deb + pH + dur + pho + nit +
amm + oxy + dbo, data = env2)

Partitioning of variance:
              Inertia Proportion
Total          0.5025     1.0000
Constrained    0.3654     0.7271
Unconstrained  0.1371     0.2729
```

```
Eigenvalues, and their contribution to the variance

Importance of components:
                         RDA1   RDA2   RDA3   RDA4   RDA5 ...
Eigenvalue              0.228 0.0537 0.0321 0.0232 0.0087 ...
Proportion Explained    0.454 0.1069 0.0639 0.0462 0.0173 ...
Cumulative Proportion   0.454 0.5607 0.6247 0.6708 0.6882 ...
                           PC1    PC2    PC3    PC4    PC5 ...
Eigenvalue             0.0458  0.0281 0.0153 0.0140 0.0098...
Proportion Explained   0.0911  0.0560 0.0304 0.0278 0.0195...
Cumulative Proportion 0.8182 0.8742 0.9047 0.9325 0.95208...

Accumulated constrained eigenvalues
Importance of components:
                         RDA1   RDA2   RDA3   RDA4   RDA5...
Eigenvalue              0.228 0.0537 0.0321 0.0232 0.0087...
Proportion Explained   0.624 0.1470 0.0879 0.0635 0.0238...
Cumulative Proportion 0.624 0.7712 0.8591 0.9226 0.9465...

Scaling 2 for species and site scores
* Species are scaled proportional to eigenvalues
* Sites are unscaled: weighted dispersion equal on all
dimensions
* General scaling constant of scores:  1.936760

Species scores

        RDA1     RDA2      RDA3      RDA4      RDA5      RDA6
CHA  0.13383  0.11623 -0.238180  0.018611  0.043221 -0.029737
TRU  0.64238  0.06648  0.123713  0.181572 -0.009691  0.029793
(...)
ANG -0.19442  0.14149  0.033659  0.017387  0.008110  0.017638

Site scores (weighted sums of species scores)

       RDA1      RDA2     RDA3      RDA4      RDA5      RDA6
1   0.40151 -0.154306  0.55539  1.600773  0.191866  0.916893
2   0.53523 -0.025084  0.43389  0.294615 -0.518456  0.458860
(...)
30 -0.48932  0.321417  0.31431  0.278218  0.487541 -0.151031

Site constraints (linear combinations of constraining variables)

       RDA1      RDA2     RDA3      RDA4      RDA5     RDA6
1   0.55135  0.002395  0.47774  0.626878 -0.210700  0.31511
2   0.29737  0.105715  0.64862  0.261161 -0.057741  0.09322
(...)
30 -0.42572  0.338080  0.24960  0.345839  0.404692 -0.13777
```

```
Biplot scores for constraining variables

                 RDA1      RDA2     RDA3     RDA4      RDA5
alt            0.8239 -0.203257  0.46604 -0.16936  0.003229
penmoderate   -0.3592 -0.008729 -0.21727 -0.18278  0.157934
(...)
dbo           -0.5171 -0.791805 -0.15652  0.22064  0.075568

Centroids for factor constraints

                 RDA1      RDA2     RDA3     RDA4     RDA5
penlow        -0.2800  0.005530 -0.09025  0.07614 -0.07860
(...)
penvery_steep  0.3908 -0.094679  0.28933  0.02307 -0.12181
```

As was the case in PCA, this output requires some explanations. Some of the results are similar to those of a PCA, but additional items are provided.

- *Partitioning of variance:* the overall variance is partitioned into constrained and unconstrained fractions. The constrained fraction is the amount of variance of the **Y** matrix explained by the explanatory variables. Expressed as a proportion, it is equivalent to an R^2 in multiple regression; in RDA this quantity is also called the bimultivariate redundancy statistic. However, *this* R^2 *is biased*, like the unadjusted R^2 of multiple regression. We present the computation of an adjusted, unbiased R^2 below.
- *Eigenvalues and their contribution to the variance:* this analysis yielded 12 canonical axes (with eigenvalues labelled RDA1 to RDA12) and 16 additional, unconstrained axes for the residuals (with eigenvalues labelled PC1 to PC16). The results give the eigenvalues themselves, as well as the cumulative proportion of variance explained (for the RDA axes) or represented (for the residual axes). The last cumulative value is therefore 1. The cumulative contribution to the variance obtained by the 12 canonical axes is the proportion of the total variance of the response data explained by the RDA. It is the same value as the "Proportion constrained" presented above; it is 0.7271 in this example.
- One feature of the eigenvalues is worth mentioning. Observe that the canonical eigenvalues RDA1 to RDA12 are (of course) decreasing in value; the first residual eigenvalue (PC1), on the contrary, is *larger* than the last canonical eigenvalue (in this example, it is actually larger than most RDA eigenvalues). This means that the first residual structure (axis) of the data has more variance than some of the structures that can be explained by the explanatory variables in **X**. It is up to the user to exploit this information, for example, by plotting the

first pair of residual axes and making hypotheses about the causes of the features revealed. These causes should not, however, involve variables that have already been used in the model, except if one suspects that products (interactions) or higher-order combinations (e.g. squared variables) may be required.

- An important distinction must be made: the canonical (RDAx) eigenvalues measure amounts of variance *explained* by the RDA model, whereas the residual (PCx) eigenvalues measure amounts of variance *represented* by the residual axes, but not explained by the RDA model.
- *Accumulated constrained eigenvalues*: these are cumulative amounts of variance expressed as proportions of the total *explained* variance, as opposed to their contribution to the *total* variance described above.
- *Species scores* are the coordinates of the tips of the vectors representing the response variables in the bi- or triplots. As in PCA, they depend on the scaling chosen.
- *Site scores (weighted sums of species scores):* coordinates of the sites as expressed in the space of the response variables **Y**.
- *Site constraints (linear combinations of constraining variables)*: coordinates of the sites in the space of the explanatory variables **X**. These are the fitted site scores.
- *Biplot scores for constraining variables*: coordinates of the tips of the vectors representing the explanatory variables. These coordinates are obtained as follows: correlations are computed between the explanatory variables and the fitted site scores, and then these correlations are transformed to produce the biplot scores. All variables, including $k-1$ levels of factors with k levels, are represented in this table. For factors, however, a representation of the centroids of the levels is preferable. See below.
- *Centroids for factor constraints*: coordinates of centroids of levels of factor variables, i.e. means of the scores of the sites possessing state "1" for a given level.

In the **`rda()`** output, an interesting information is missing: the canonical coefficients, i.e. the equivalent of regression coefficients for each explanatory variable on each canonical axis. These coefficients can be retrieved by typing **`coef()`**:

```
# How to obtain canonical coefficients from an rda() object
coef(spe.rda)
```

Hint *Type* **`?coef.cca`** *and see how to obtain fitted and residual values. There is also a* **`calibrate()`** *function allowing the projection of new sites into a canonical ordination result for bioindication purposes, although with some conditions.*

6.3.2.3 Retrieving, Interpreting and Plotting Results from a `vegan` RDA Output Object

The various elements making up the **`rda()`** output object can be retrieved in the same way as for a PCA. This is useful when you want to use the results outside the functions provided by **`vegan`** to handle them.

As mentioned above, the R^2 *of a RDA is biased* like the ordinary R^2 of multiple regression, and for the same reason (Peres-Neto et al. 2006). On the one hand, any variable included in explanatory matrix **X** increases the R^2, irrespective of it being related, or not, to the response data. On the other hand, the accumulation of explanatory variables inflates the apparent amount of explained variance because of random correlations. This problem can be cured by *adjusting the* R^2 using Ezekiel's formula (Ezekiel 1930), which is also valid in the multivariate case:

$$R^2_{\text{adj}} = 1 - \frac{n-1}{n-m-1}(1-R^2) \tag{6.1}$$

where n is the number of objects and m is the number of explanatory variables (or, more precisely, the number of degrees of freedom of the model). Ezekiel's adjustment can be used as long as the number of degrees of freedom of the model is not overly large with respect to the number of observations. As a rule of thumb, this adjustment may be overly conservative when $m > n/2$. An adjusted R^2 near 0 indicates that **X** does not explain more of the variation of **Y** than random normal variables would do. Adjusted R^2 values can be negative, indicating that the explanatory variables **X** do worse than random normal variables would.

In our example, $n = 29$ and $m = 12$ (remember that one of the ten variables is a factor with $k = 4$ levels, so it occupies 3 degrees of freedom). The R^2 and adjusted R^2 can be computed using **vegan**'s function **`RsquareAdj()`**.

```
# Retrieval of the adjusted R^2
# ****************************

# Unadjusted R^2 retrieved from the rda result
(R2 <- RsquareAdj(spe.rda)$r.squared)

# Adjusted R^2 retrieved from the rda object
(R2adj <- RsquareAdj(spe.rda)$adj.r.squared)
```

```
  # As one can see, the adjustment has substantially reduced
  # the value of the R^2. The adjusted R^2 measures the
  # unbiased amount of explained variation and will be used
  # later for variation partitioning.
```

Let us now plot the results of our RDA (Fig. 6.1). We can call this a triplot since there are three different entities in the plot: sites, response variables and explanatory variables. To differentiate the latter two, we draw arrowheads only on the vectors of the quantitative explanatory variables, not on the response variable vectors.

```
# Triplots of the rda results
# ***************************
# Scaling 1: distance triplot
plot(spe.rda, scaling=1, main="Triplot RDA spe.hel ~ env2 -
scaling 1 - wa scores")

  # This plot displays all our entities: sites, species,
  # explanatory variables as arrows (with heads), or centroids,
  # depending on their type. But arrows for species are missing.
  # Let us add them without heads to make them appear different
  # from the explanatory variables, after retrieval from the
  # output object:

spe.sc <- scores(spe.rda, choices=1:2, scaling=1, display="sp")
arrows(0, 0, spe.sc[,1], spe.sc[,2], length=0, lty=1, col="red")

# Scaling 2 (default): correlation triplot
plot(spe.rda,
 main="Triplot RDA spe.hel ~ env2 - scaling 2 - wa scores")
spe2.sc <- scores(spe.rda, choices=1:2, display="sp")
arrows(0, 0, spe2.sc[,1], spe2.sc[,2], length=0, lty=1,
 col="red")
```

Hint *In the* **scores()** *function argument* `choices=` *indicates which axes are to be selected. Be careful to specify the scaling if it is different from 2.*

The two RDA triplots use the site scores that are weighted sums of species (abbreviated "`wa`" in **vegan**). The choice between these and the fitted site scores (abbreviated "`lc`") for the triplots is still controversial. On the one hand, the fitted site scores are strictly orthogonal linear combinations of the explanatory variables; but they nevertheless represent clearly and exclusively what can be modelled using the explanatory variables at hand. On the other hand, the site scores that are weighted sums of species

appear more robust to noise in the environmental variables: McCune (1997) showed that if the latter contain much random error, the resulting lc plots may be completely scrambled. However, weighted sums of species (wa) scores are "contaminated" scores, halfway between the model fitted by the RDA procedure and the original data, and as such it is not entirely clear how they should be interpreted. When RDA is used as a form of analysis of variance (Sect. 6.3.2.8), all replicate sites with the same combination of factor levels are represented on top of one another in the fitted site scores (lc) triplot; the weighted sums of species (wa) triplot is preferable in that case because the sites are separated in the plot and their labels can be read.

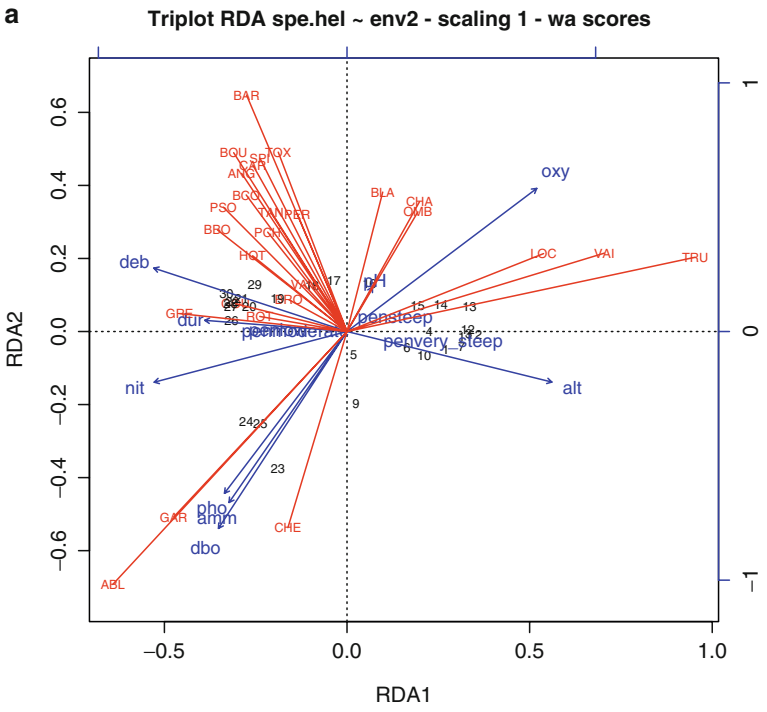

Fig. 6.1 (**a**) RDA triplot of the Hellinger-transformed Doubs fish abundance data constrained by all environmental variables, scaling 1. The bottom and left-hand scales are for the objects and the response variables, the top and right-hand scales are for the explanatory variables

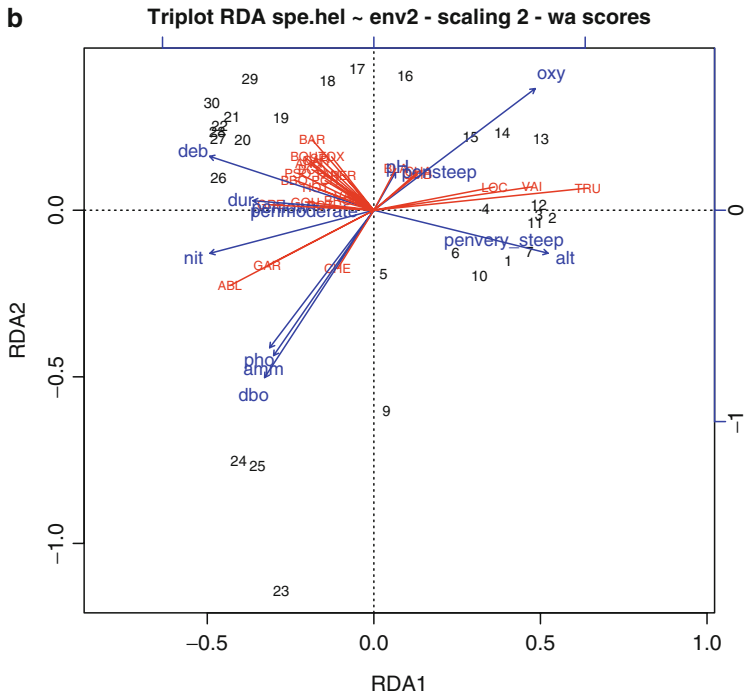

Fig. 6.1 (continued) (**b**) RDA triplot of the Hellinger-transformed Doubs fish abundance data constrained by all environmental variables, scaling 2. The bottom and left-hand scales are for the objects and the response variables, the top and right-hand scales are for the explanatory variables

Independently of the choice of site scores, the interpretation of the constrained triplots must be preceded by a test of statistical significance (see below). As in multiple regression, a non-significant result must not be interpreted and must be discarded.

Since the wa–lc issue is open, let us also represent our triplots using the fitted site scores.

```
# Site scores as linear combinations of the environmental
# variables

# Scaling 1
plot(spe.rda, scaling=1, display=c("sp","lc","cn"),
  main="Triplot RDA spe.hel ~ env2 - scaling 1 - lc scores")
arrows(0, 0, spe.sc[,1], spe.sc[,2], length=0, lty=1, col="red")

# Scaling 2
plot(spe.rda, display=c("sp","lc","cn"),
  main="Triplot RDA spe.hel ~ env2 - scaling 2 - lc scores")
arrows(0, 0, spe2.sc[,1], spe2.sc[,2], length=0, lty=1,
  col="red")
```

Hint *See how to choose the elements to be plotted, using the argument* `display=c(...)`. *In this argument, "*`sp`*" stands for species, "*`wa`*" for site scores in the species space (weighted averages), "*`lc`*" for fitted site scores (linear combinations of explanatory variables), and "*`cn`*" for constraints (i.e. the explanatory variables).*

For the species and sites, the interpretation of the two scalings is the same as in PCA. However, the presence of vectors and centroids of explanatory variables calls for *additional interpretation rules*. Here are the essential ones (see Legendre and Legendre 1998, pp. 586–587):

- *Scaling 1* – distance biplot: (1) Projecting an object at right angle on a response variable **or a quantitative explanatory variable** approximates the position of the object along that variable. (2) The angles **between response and explanatory variables** in the biplot reflect their correlations (*but not the angles among response variables*). (3) The relationship between the **centroid** of a qualitative explanatory variable and a response variable (species) is found by projecting the centroid at right angle on the species variable, as for individual objects, since we are projecting the centroid of a group of objects. (4) Distances among **centroids**, and between **centroids** and individual objects, approximate their Euclidean distances.
- *Scaling 2* – correlation biplot: (1) Projecting an object at right angle on a response **or a quantitative explanatory variable** approximates the value of the object along that variable. (2) The angles in the biplot between response and **explanatory variables**, and between response variables themselves or **explanatory variables themselves**, reflect their correlations. (3) The relationship between the **centroid** of a qualitative explanatory variable and a response variable (species) is found by projecting the centroid at right angle on the species

variable (as for individual objects). (4) *Distances among* ***centroids****, and between centroids and individual objects, do not approximate their Euclidean distances.*

On these bases, it is now possible to interpret the triplots. Let us take as examples the pair of plots representing the fitted site scores (Fig. 6.2). The numerical output shows that the first two canonical axes explain together 56.1% of the total variance of the data, the first axis alone explaining 45.4%. These are unadjusted values, however. Since $R^2_{adj} = 0.5224$, the percentages of accumulated constrained eigenvalues show that the first axis alone explains $0.5224 \times 0.6242 = 0.326$ or 32.6% variance, and the two first $0.5224 \times 0.7712 = 0.4029$ or 40.3% variance. We can be confident that the major trends have been modelled in this analysis. Because ecological data are generally quite noisy, one should never expect to obtain a very high value of R^2_{adj} . Furthermore, the first unconstrained eigenvalue (PC1) is comparatively small, which means that it does not display any important residual structure of the response data.

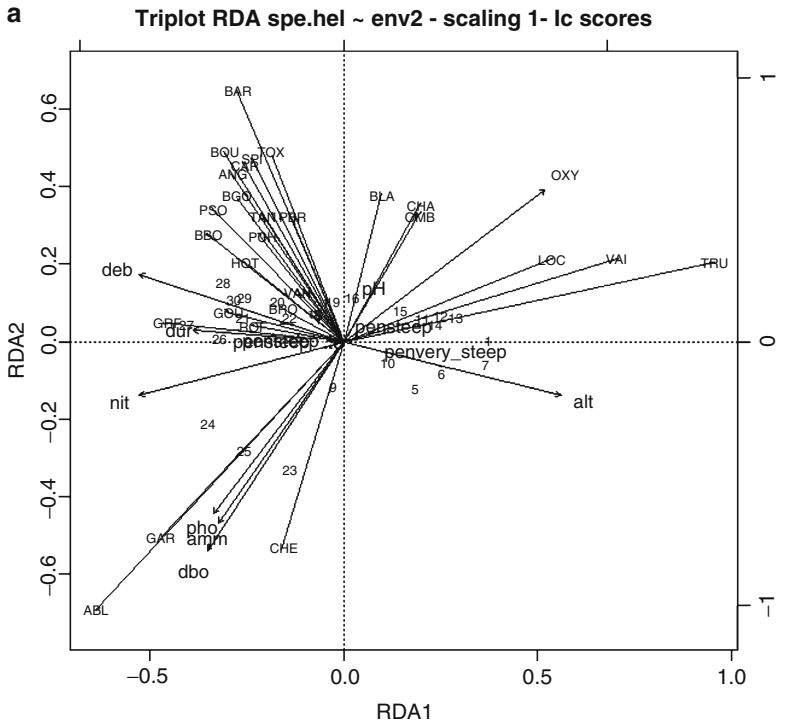

Fig. 6.2 (**a**) Triplot of the RDA with fitted site scores and all environmental variables. Scaling 1

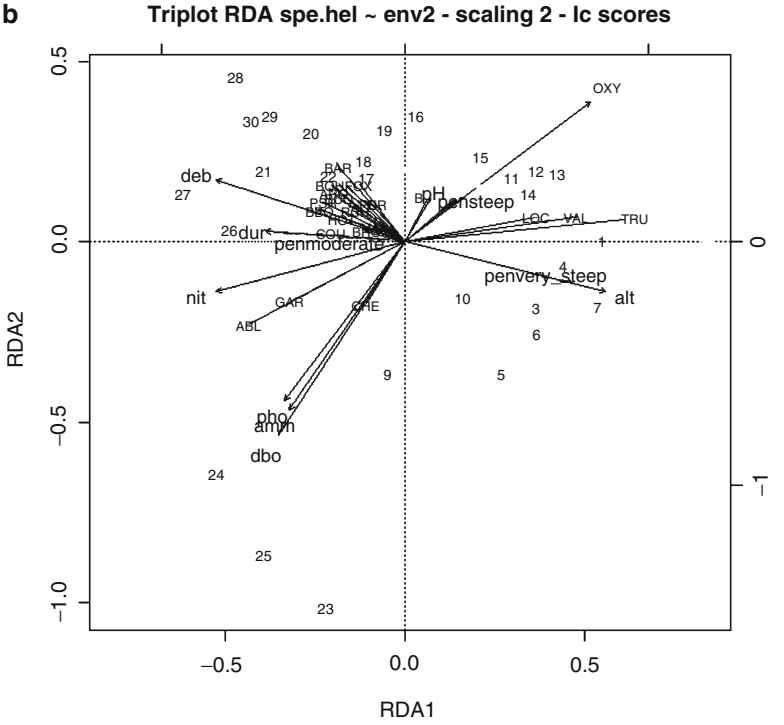

Fig. 6.2 (continued) (**b**) Triplot of the RDA with fitted site scores and all environmental variables. Scaling 2

These triplots show that oxygen (`oxy`), altitude (`alt`), nitrates (`nit`) and discharge (`deb`), as well as slope (mainly the level `penvery_steep`) play an important role in the dispersion of the sites along the first axis. Both triplots oppose the upper and lower parts of the river along the first axis. The scaling 2 triplot shows three groups of fish species correlated with different sets of explanatory variables: the brown trout (`TRU`), Eurasian minnow (`VAI`) and stone loach (`LOC`) are found in the first half of the sites, and are correlated with high oxygen content and slope as well as higher altitude. The bleak (`ABL`), roach (`GAR`) and European chub (`CHE`), on the opposite, are related to sites 23, 24 and 25 characterized by high phosphates (`pho`), ammonium (`amm`) and biological oxygen demand (`dbo`) levels. Most other species are bunched together away from these extremes. They show mostly shorter projections, indicating that they are either present over most portions of the river or related to intermediate ecological conditions.

6.3.2.4 Permutation Tests of RDA Results

Due to widespread problems of non-normal distributions in ecological data, classical parametric tests are often not appropriate in this field. This is why most methods of ecological data analysis nowadays resort to permutation tests whenever possible. The principle of a permutation test is to generate a reference distribution of the chosen statistic under the null hypothesis H_0 by randomly permuting appropriate elements of the data a large number of times and recomputing the statistic each time. Then, one compares the true value of the statistic to this reference distribution. The p value is computed as the proportion of the permuted values equal to or larger than the true (unpermuted) value of the statistic for a one-tailed test in the upper tail, like the F test used in RDA. The true value is included in this count. The null hypothesis is rejected if this p value is equal to or smaller than the predefined significance level α.

Three elements are critical in the construction of a permutation test: (1) the choice of the permutable units, (2) the choice of the statistic and (3) the permutation scheme.

The *permutable units* are often the response data (random permutation of the rows of the **Y** data matrix), but sometimes other permutable units must be defined. See Legendre and Legendre (1998, pp. 606 *et sq*. and especially Table 11.7, p. 612). In the simple case presented here, the null hypothesis H_0 states (loosely) that no (linear) relationship exists between the response data **Y** and the explanatory variables **X**. In terms of permutation tests, this means that the sites in either matrix can be permuted randomly to produce realizations of this H_0, thereby destroying the possible relationship between a given fish assemblage and the ecological condition of its site. A permutation test does this 100, 1000 or 10000 times to produce a large sample of test statistics to which the true value is compared.

In RDA, the use of parametric tests is possible only when the response variables are standardized (Miller 1975; Legendre et al. 2011), which is inappropriate for community composition data for example. So all RDA programs for ecologists implement permutation tests. The test statistic (often called *pseudo-F*) is defined as follows:

$$F = \frac{\mathrm{SS}(\hat{\mathbf{Y}})/m}{\mathrm{RSS}/(n-m-1)} \tag{6.2}$$

where m is the number of canonical eigenvalues (or degrees of freedom of the model), SS($\hat{\mathbf{Y}}$) (explained variation) is the sum of squares of the table of fitted values and residual sum of squares (RSS) is the total sum-of-squares of **Y**, SS(**Y**), minus the explained variation SS($\hat{\mathbf{Y}}$).

The test of significance of individual axes is based on the same principle. The first canonical eigenvalue is tested as follows: that eigenvalue is the numerator of the F statistic, whereas the SS(**Y**) minus that eigenvalue/$(n-1-1)$ is the denominator (since $m=1$ in this case). The test of the subsequent canonical axes is more complicated: the previously tested canonical axes have to be included as covariables in the analysis, as in Sect. 6.3.2.5, and the RDA is recomputed; see Legendre et al. 2011 for details.

The *permutation scheme* describes how the units are permuted. In most cases, the permutations are free, i.e. all units are considered equivalent and fully exchangeable, and the data rows of **Y** are permuted. In partial canonical analysis (next subsection), however, it is the residuals of some regression model that are permuted. Finally in some situations, permutations may be restricted within subgroups of data, for instance, when a multistate covariable or a multilevel experimental factor is included in the analysis.

With this in mind, we can now test our RDA results. Let us run a global test first, followed by a test of the canonical axes. The test function is called **`anova()`**. This name is unfortunate, since it leads to confusion with the classical ANOVA test, which it is not.

```
# Global test of the RDA result
anova.cca(spe.rda, step=1000)
# Tests of all canonical axes
anova.cca(spe.rda, by="axis", step=1000)

  # The test of the axes can only be run with the formula
  # interface. How many canonical axes are significant?
```

Hint *Argument* `"step"` *gives the minimal number of permutations requested to assess if the F value of a test is obviously significant or not.*

Of course, given that these tests are available, it is useless to apply other criteria like the broken-stick or Kaiser–Guttman's criterion to canonical axes. These could be applied to the residual, unconstrained axes, however. Let us apply the Kaiser–Guttman criterion:

```
# Apply Kaiser-Guttman criterion to residual axes
spe.rda$CA$eig[spe.rda$CA$eig > mean(spe.rda$CA$eig)]

  # Apparently there is still some interesting variation in these
  # data that has not been explained by our set of environmental
  # variables
```

6.3.2.5 Partial RDA

Partial canonical ordination is the multivariate equivalent of partial linear regression. For instance, it is possible to run an RDA of a (transformed) plant species data matrix **Y**, explained by a matrix of climatic variables **X**, in the presence of soil covariables **W**. Such an analysis would allow the user to display the patterns of the species data uniquely explained by a linear model of the climatic variables when the effect of the soil factors is held constant.

We will now run an example using the Doubs data. At the beginning of this chapter, we created two objects containing subsets of environmental variables. One subset contains physiographical variables (`envtopo`), i.e. altitude, slope (the original, quantitative variable, not the four-level factor used in the previous RDA) and discharge; the other contains variables describing water chemistry (`envchem`), i.e. pH, hardness, phosphates, nitrates, ammonium, oxygen content as well as biological oxygen demand. The analysis that follows determines whether water chemistry significantly explains the fish species patterns when the effect of the topographical gradient is held constant.

In **vegan**, partial RDA can be run in two ways, depending on whether one uses the simple or the formula interface. In the first cases, explanatory variables and covariables may or may not be in separate objects, which can belong to classes vector, matrix, or data.frame; factor variables cannot be used with that notation, however. With the formula interface, the **X** and **W** data sets must be in the same object, which must be a data frame and may contain factor variables.

```
# Partial RDA: effect of water chemistry, holding physiography
# constant

# Simple interface; X and W may be separate tables of
# quantitative variables
spechem.physio <- rda(spe.hel, envchem, envtopo)
spechem.physio
summary(spechem.physio)

# Formula interface; X and W must be in the same data frame
class(env)
spechem.physio2 <- rda(spe.hel ~ pH + dur + pho + nit + amm +
  oxy + dbo + Condition(alt + pen + deb), data = env)
spechem.physio2

  # The results of the two analyses are identical.
```

Hint *The formula interface may seem cumbersome, but it allows a better control of the model and the use of factors among the constraints (**X**) or the conditions (**W**). One could therefore have used the factor-transformed pen variable in the second analysis, but not in the first one.*

Here again, some additional explanations are needed about the summary output.

- *Partitioning of variance*: This item now shows four components. The first one (`Total`) is, as usual, the total inertia (variance in this case) of the response data. The second line (`Conditioned`) gives the amount of variance that has been explained by the covariables and removed.[2] The third line (`Constrained`) gives the amount of variance uniquely explained by the explanatory variables. The fourth line (`Unconstrained`) gives the residual variance. *Beware:* the values given as proportions (right-hand column) are *unadjusted* and are therefore biased. For a proper computation of unbiased, adjusted R^2 and partial R^2, see below (variation partitioning).
- *Eigenvalues, and their contribution to the variance after removing the contribution of conditioning variables:* these values and proportions are *partial* in the sense that the effects of the covariables have been removed. The sum of all these eigenvalues corresponds therefore to the sum of the constrained and unconstrained (residual) variances, excluding the conditioned (i.e. removed) variance.

We can now test the partial RDA and, if it is significant, draw triplots of the first pair of axes (Fig. 6.3).

```
# Test of the partial RDA (using the results with the formula
# interface to allow the tests of the axes to be run)
anova.cca(spechem.physio2, step=1000)
anova.cca(spechem.physio2, step=1000, by="axis")

# Partial RDA triplots (with fitted site scores)

# Scaling 1
plot(spechem.physio, scaling=1, display=c("sp","lc","cn"),
  main="Triplot RDA spe.hel ~ chem | Topo - scaling 1 - lc
  scores")
spe3.sc <- scores(spechem.physio, choices=1:2, scaling=1,
  display="sp")
arrows(0, 0, spe3.sc[,1], spe3.sc[,2], length=0, lty=1,
col="red")
```

[2] Mathematically, to partial out the effect of a matrix **W** from an ordination of **Y** by **X**, one computes the residuals of a multivariate multiple regression of **X** on **W** and uses these residuals as the explanatory variables.

```
# Scaling 2
plot(spechem.physio, display=c("sp","lc","cn"),
 main="Triplot RDA spe.hel ~ chem | Topo - scaling 2 - lc
 scores")
spe4.sc <- scores(spechem.physio, choices=1:2, display="sp")
arrows(0, 0, spe4.sc[,1], spe4.sc[,2], length=0, lty=1,
 col="red")
```

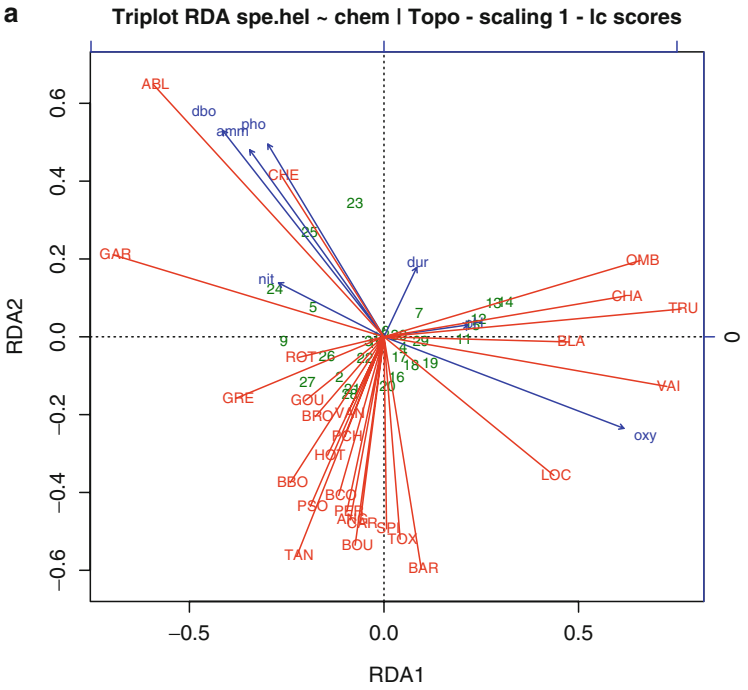

Fig. 6.3 (**a**) RDA triplot, Doubs Hellinger-transformed fish data explained by chemistry, controlling for physiography. Sites: model scores (option "`lc`" of argument `display`). Scaling 1

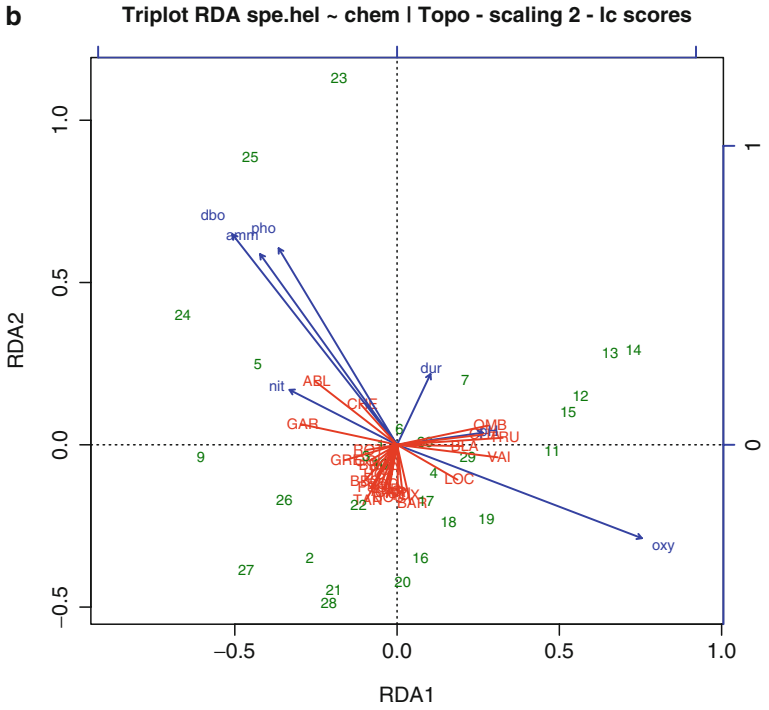

Fig. 6.3 (continued) (**b**) RDA triplot, Doubs Hellinger-transformed fish data explained by chemistry, controlling for physiography. Sites: model scores (option "`lc`" of argument `display`). Scaling 2

As could be expected, the results of this partial analysis differ somewhat from the previous results, but not fundamentally. Although there are interesting features to discuss about the amounts of variance explained, we postpone this aspect until we have seen a quick and elegant manner to compute the adjusted R^2 of partial analyses, by means of variation partitioning (Sect. 6.3.2.7). On the triplots, the explanatory variables show the same relationships to one another, but some of them (hardness (`dur`) and nitrates (`nit`)) are less important to explain the fish community structure, as shown by their shorter vectors. This may be due to the fact that these two variables are well correlated with the position along the river, and therefore their apparent effect on the fish community may have been spurious and has

been removed by the analysis, which controlled for the effect of the physiographical variables. The scaling 1 triplot shows that the sites are not as cleanly ordered by their succession along the river. This indicates that the chemical variables do not necessarily follow that order and that the fish community responds significantly to the chemical constraints irrespective of their locations along the river.

6.3.2.6 Forward Selection of Explanatory Variables

It happens sometimes that one wishes to reduce the number of explanatory variables. The reasons vary: search for parsimony, rich data set but poor *a priori* hypotheses, or a method producing a large set of explanatory variables which must be reduced afterwards (as in eigenvector-based spatial analysis, see Chap. 7). In the Doubs data, there could be two reasons (albeit not compelling) to reduce the number of explanatory variables: search for parsimony, and possible strong linear dependencies (correlations) among the explanatory variables in the RDA model, which could render the regression coefficients of the explanatory variables in the model unstable.

Linear dependencies can be explored by computing the variables' variance inflation factors (VIF), which measure the proportion by which the variance of a regression coefficient is inflated in the presence of other explanatory variables. VIFs above 20 indicate strong collinearity. Ideally, VIFs above 10 should be at least examined, and avoided if possible. VIFs can be computed in **vegan** after RDA or CCA:

```
# Variance inflation factors (VIF) in two RDAs
# ********************************************
# First RDA of this Chapter: all environmental variables
vif.cca(spe.rda)
vif.cca(spechem.physio)  # Partial RDA
```

Several VIF values are above 10 or even 20 in these analyses so that a reduction of the number of explanatory variables is justified.

No single, perfect method exists for variable reduction, besides the examination of all possible subsets of explanatory variables, which is time-prohibitive for real data sets. In multiple regression, the three usual methods are forward, backward and stepwise selection of explanatory variables, the latter one being a combination of the former two. In RDA, forward selection is the method most often applied. This method works as follows:

- Compute RDA of the response data with all explanatory variables in turn.
- Two sets of criteria are used in the presently available functions for selecting the "best" explanatory variable: (1) in the **`forward.sel()`** function of **`packfor`** (see below), one selects the explanatory variable with the highest R^2 if that variable is also significant (permutation test) at a preselected significance level; "best" refers to the variable that explains the largest portion of the variance of the response data. (2) In the **`ordistep()`** function of **`vegan`** (see below), the significance of the F-statistics associated with all variables is tested using permutation tests, and the most significant explanatory variable is selected. In case of equality, the variable that has the lowest value of the Akaike Information Criterion (AIC) is selected for inclusion in the model; "best" refers here to the most significant variable.
- The next task is to look for a second (third, fourth, etc.) variable to include in the explanatory model. Compute all models containing the previously selected variable(s) plus one of the remaining explanatory variables. (1) In **`forward.sel()`**, select the new variable that forms the model with the highest R^2 (which is the same as selecting the variable with the highest semipartial R^2) if the partial contribution of that variable is significant at the preselected significance level; the test of significance is a permutation test by partial RDA. (2) In **`ordistep()`**, select the new variable whose partial contribution is the most significant. In case of ties, select the variable with the lowest AIC.
- The process continues until no more significant variable can enter the model.

In its original form, forward selection in **`packfor`**'s **`forward.sel()`** used the preselected significance level α as the main stopping criterion: selection was stopped when no additional variable had a p value smaller than or equal to the significance level. However, this criterion is known to be overly liberal, either by selecting sometimes a "significant" model when none should have been identified (hence inflating type I error), or by including too many explanatory variables into the model (hence inflating the amount of explained variance). Blanchet et al. (2008a) addressed this double problem and proposed solutions to improve this technique:

– To prevent the problem of inflation of the overall type I error, a global test using all explanatory variables is first run. If, and only if, that test is significant, the forward selection is performed.
– To reduce the risk of incorporating too many variables into the model, the adjusted coefficient of multiple determination R^2_{adj} of the global model (containing all the potential explanatory variables) is computed, and used as a second stopping criterion. Forward selection is stopped if one of the following criteria is reached: the traditional significance level α or the global R^2_{adj}; in other words,

if a candidate variable is deemed non-significant or if it brings the R^2_{adj} of the current model over the value of the R^2_{adj} of the global model.

This new, enhanced procedure has been implemented in the **packfor** package of Dray et al. (2007). The function is called **forward.sel()**. We can apply it to the Doubs data. This procedure does not allow for factor variables; we will use the `env` data set in the analysis.

```
# Forward selection of explanatory variables
# using a double stopping criterion (Blanchet et al. 2008a)

# 1. RDA with all explanatory variables
spe.rda.all <- rda(spe.hel ~ ., data=env)

# 2. Global adjusted R^2
(R2a.all <- RsquareAdj(spe.rda.all)$adj.r.squared)

# 3. Forward selection using packfor's forward.sel()
# library(packfor)  # if not already loaded
forward.sel(spe.hel, env, adjR2thresh=R2a.all)

 # Note: as mentioned in the informative message printed by the
 # function, the last variable in the selection list is the one
 # that violates the adjR2thresh stopping criterion. It should
 # not be included in the selection.
```

The result shows that one can devise an explanatory model of the fish community that is much more parsimonious than the model involving all explanatory variables. Three variables are enough to obtain an R^2_{adj} almost as large as that of the global model. Besides, observe that the **forward.sel()** selection has been stopped *after* the variable leading to a value exceeding the R^2_{adj} has been entered. A moderate excess (here, 0.5948 versus 0.5864) is certainly not a great problem. The selection diagnostics show that the last variable entered (`pen`) increases the R^2_{adj} by 5.5%.

Another function performs several kinds of variable selection: **vegan**'s **ordistep()**. This function does not implement Blanchet et al. (2008a)'s second, R^2_{adj}-based stopping criterion. So this criterion must be applied by hand to the result in the way presented below. **ordistep()** selects the variables on the basis of their permutational p values, and on AIC in case of ties.

```
# Forward selection using vegan's ordistep().
# This function allows the use of factors. Options are also
# available for stepwise and backward selection of the
```

```
# explanatory variables.
step.forward <- ordistep(rda(spe.hel ~ 1, data=env),
scope=formula(spe.rda.all), direction="forward", pstep=1000)
```

The selected variables are the same in this example as when using **`forward.sel()`** without the `adjR2thresh` stopping criterion of Blanchet et al. (2008a). One could apply that stopping criterion by computing the R^2_{adj} of RDAs that incorporate the variables in their order of inclusion by **`ordistep()`** and checking when the cumulative R^2_{adj} criterion of 0.586435 is exceeded:

```
RsquareAdj(rda(spe.hel ~ alt, data=env))$adj.r.squared
RsquareAdj(rda(spe.hel ~ alt+oxy, data=env))$adj.r.squared
RsquareAdj(rda(spe.hel ~ alt+oxy+dbo, data=env))$adj.r.squared
RsquareAdj(rda(spe.hel ~ alt+oxy+dbo+pen,
 data=env))$adj.r.squared
```

These analyses show that the most parsimonious attitude would be to settle for a model containing only three explanatory variables: altitude, oxygen and biological oxygen demand. What would such an analysis look like?

```
# Parsimonious RDA
# ****************

spe.rda.pars <- rda(spe.hel ~ alt + oxy + dbo, data=env)
spe.rda.pars
anova.cca(spe.rda.pars, step=1000)
anova.cca(spe.rda.pars, step=1000, by="axis")
vif.cca(spe.rda.pars)
(R2a.pars <- RsquareAdj(spe.rda.pars)$adj.r.squared)
```

These results are fascinating in that they demonstrate how parsimony can help improve the quality of a model. With a moderate cost in explanatory power, we produced a model that is as highly significant, has no harmful collinearity (all VIFs are now well below 10), and can be decomposed into *three significant* canonical axes. The global model produced only two significant axes.

It is now time to produce triplots of this result (Fig. 6.4). We compare them to the triplots of the global analysis.

```
# Triplots of the parsimonious RDA (with fitted site scores)
# ***********************************************************

# Scaling 1
plot(spe.rda.pars, scaling=1, display=c("sp","lc","cn"),
  main="Triplot RDA spe.hel ~ alt+oxy+dbo - scaling 1 - lc
  scores")
spe4.sc = scores(spe.rda.pars, choices=1:2, scaling = 1,
  display="sp")
arrows(0, 0, spe4.sc[,1], spe4.sc[,2], length=0, lty=1,
  col="red")

# Scaling 2
plot(spe.rda.pars, display=c("sp","lc","cn"),
  main="Triplot RDA spe.hel ~ alt+oxy+dbo - scaling 2 - lc
  scores")
spe5.sc = scores(spe.rda.pars, choices=1:2, display="sp")
arrows(0, 0, spe5.sc[,1], spe5.sc[,2], length=0, lty=1,
  col="red")

  # Since there is now a third significant canonical axis, you
  # could plot other combinations: axes 1 and 3, axes 2 and 3.
```

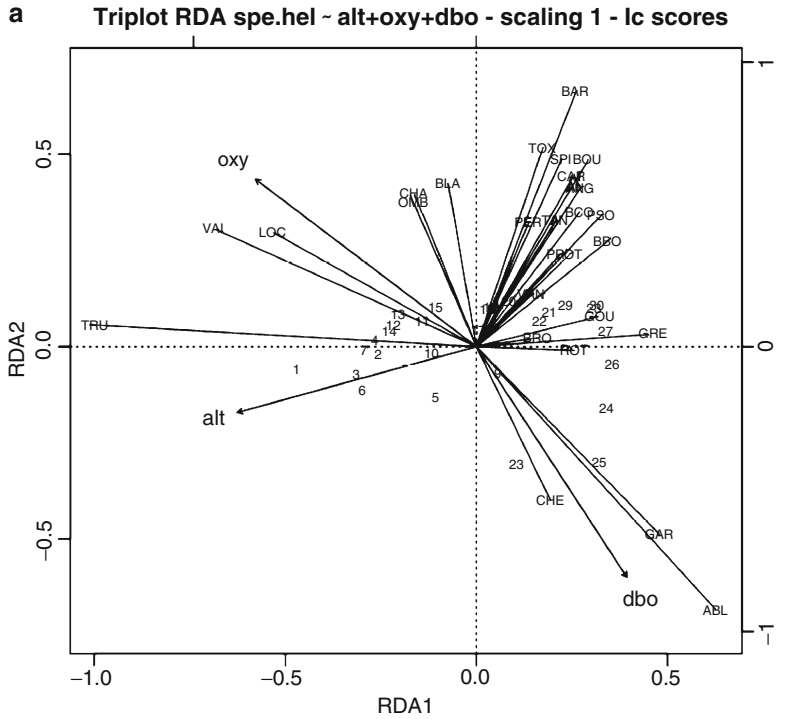

Fig. 6.4 (**a**) RDA triplot, Hellinger-transformed Doubs fish data constrained by three environmental variables. Model scores. Scaling 1

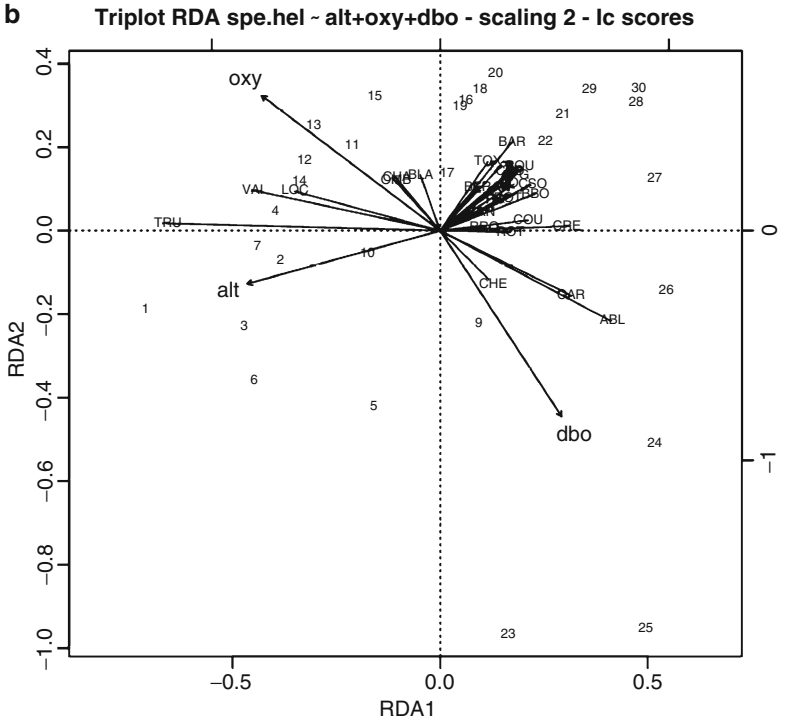

Fig. 6.4 (continued) (**b**) RDA triplot, Hellinger-transformed Doubs fish data constrained by three environmental variables. Model scores. Scaling 2

These triplots indeed present the same structures as the ones produced with all explanatory variables. We shall revisit this result later.

6.3.2.7 Variation Partitioning

A common occurrence in ecology is that one has two or more sets of explanatory variables pertaining to different classes. In the Doubs data, we have already split the environmental variables into a first subset of physiographic and a second subset of chemical variables. For various reasons, one might be interested not only in a partial analysis like the one that we conducted above, but in quantifying the variation explained by all subsets of the variables when controlling for the effect of the other subsets. In multivariate ecological analysis, a procedure of variation partitioning has been proposed to that effect by Borcard et al. (1992) and improved by the use of adjusted R^2 by Peres-Neto et al. (2006). When two explanatory data sets **X** and **W** are used, the total variation of **Y** is partitioned as in Fig. 6.5.

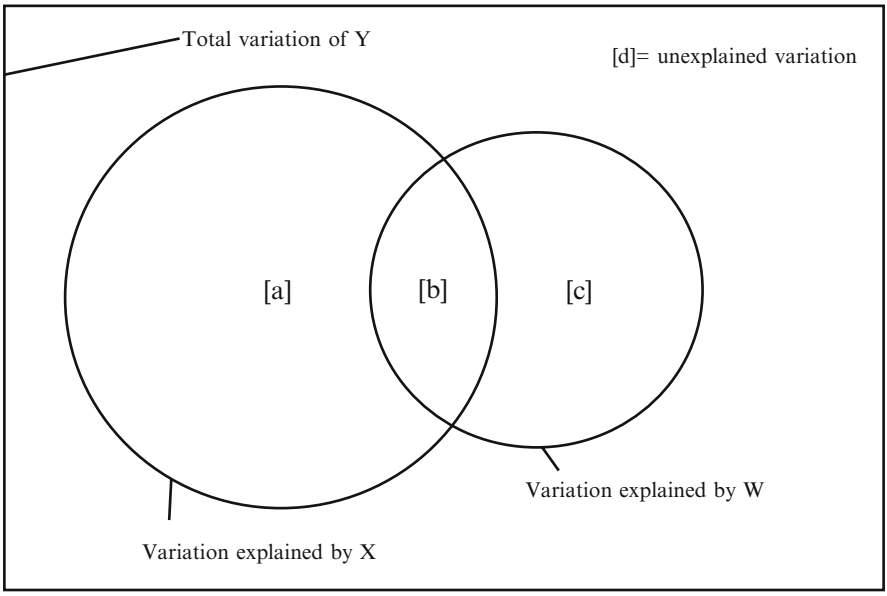

Fig. 6.5 Venn diagram of the variation partitioning of a response data set **Y** explained by two data sets **X** and **W**. The rectangle represents the total sum-of-squares of **Y**

X and **W** both explain some variation of the response data. Since the explanatory data sets are generally not orthogonal to one another (except in some cases addressed later), some amount of variation is explained jointly by the two sets (fraction [b] of Fig. 6.5). Consequently, the variation explained by all variables together is less than the sum of the variations explained by the various subsets. This is why one can express the variation explained by **X** as fraction [a+b], the variation explained by **W** as fraction [b+c] and the unexplained variation as fraction [d]. In the case of two explanatory data sets, three RDAs are needed to partition the variation into the four individual fractions [a], [b], [c] and [d].

The conceptual steps are the following:

- If necessary, forward-select the explanatory variables *separately* in each subset.
- Run an RDA of the response data **Y** by **X**. This yields fraction [a+b].
- Run an RDA of the response data **Y** by **W**. This yields fraction [b+c].
- Run an RDA of the response data **Y** by **X** and **W** together. This yields fraction [a+b+c].
- **Compute the adjusted R^2 (R^2_{adj}) of the three RDAs above**.
- Compute the fractions of adjusted variation by subtraction:
  - fraction $[a]_{adj} = [a+b+c]_{adj} - [b+c]_{adj}$
  - fraction $[c]_{adj} = [a+b+c]_{adj} - [a+b]_{adj}$

- fraction $[b]_{adj} = [a+b]_{adj} - [a]_{adj} = [b+c]_{adj} - [c]_{adj}$
- fraction $[d]_{adj} = 1 - [a+b+c]_{adj}$

The three RDAs can be tested as usual, and fractions [a] and [c] can be computed and tested by means of partial RDA. Fraction [b], however, is not an adjusted component of variance and cannot be estimated and tested by regression methods. It has zero degree of freedom. Note also that there is no equation for computing an adjusted R^2 directly for a partial RDA. The subtractive procedure described above avoids this difficulty. It has been shown by Peres-Neto et al. (2006) to produce unbiased estimates of the fractions of variation. Remember also that adjusted R^2 can be negative. See Sect. 6.3.2.3. Negative R^2_{adj} can be ignored (considered as null) for the ecological interpretation of the results.

The whole procedure of variation partitioning (except the preliminary stage of forward selection) can be run in one **R** command with up to four explanatory matrices. The function to use, available in **`vegan`**, is **`varpart()`**. Let us apply it to the Doubs data and follow up with tests of all testable fractions.

```
# Variation partitioning with two sets of explanatory variables
# ************************************************************

# Explanation of fraction labels
par(mfrow=c(1,3))
showvarparts(2) # Two explanatory matrices
showvarparts(3) # Three explanatory matrices
showvarparts(4) # Four explanatory matrices

# 1. Variation partitioning with all explanatory variables

spe.part.all <- varpart(spe.hel, envchem, envtopo)
spe.part.all
plot(spe.part.all, digits=2)

  # The plot gives correct values of the adjusted R squares, but
  # the sizes of the circles in the Venn diagram are not to
  # scale.
```

This first partitioning shows that both sets of explanatory variables contribute to the explanation of the species data. The unique contribution of the chemical variables (fraction [a], $R^2_{adj} = 0.241$) is much larger than that of physiography (fraction [c], $R^2_{adj} = 0.112$). The variation explained jointly by the two sets (fraction [b], $R^2_{adj} = 0.233$) is also large. This indicates that the chemical and physiographic

variables are intercorrelated. This is a good reason to make an effort towards parsimony, and to combine variation partitioning with forward selection.

```
# 2. Separate forward selection in each subset of environmental
#    variables

spe.chem <- rda(spe.hel, envchem)
R2a.all.chem <- RsquareAdj(spe.chem)$adj.r.squared
forward.sel(spe.hel,envchem, adjR2thresh = R2a.all.chem,
 nperm=9999)

spe.topo <- rda(spe.hel, envtopo)
R2a.all.topo <- RsquareAdj(spe.topo)$adj.r.squared
forward.sel(spe.hel,envtopo, adjR2thresh = R2a.all.topo,
 nperm=9999)

# Parsimonious subsets of explanatory variables (based on
# forward selections)
envchem.pars <- envchem[,c(4,6,7)]
envtopo.pars <- envtopo[,c(1,2)]

# Variation partitioning
(spe.part <- varpart(spe.hel, envchem.pars, envtopo.pars))
plot(spe.part, digits=2)

# Tests of all testable fractions
# Test of fractions [a+b]
anova.cca(rda(spe.hel, envchem.pars), step=1000)
# Test of fractions [b+c]
anova.cca(rda(spe.hel, envtopo.pars), step=1000)
# Test of fractions [a+b+c]
env.pars <- cbind(envchem.pars, envtopo.pars)
anova.cca(rda(spe.hel, env.pars), step=1000)
# Test of fraction [a]
anova.cca(rda(spe.hel, envchem.pars, envtopo.pars), step=1000)
# Test of fraction [c]
anova.cca(rda(spe.hel, envtopo.pars, envchem.pars), step=1000)

 # Are any of these components nonsignificant?
```

As one could have expected, forward-selecting the explanatory variables *independently* in each subset (chemistry and physiography) does nothing to prevent inter-set correlations; some of the variables retained in each set are correlated with those of the other set. Therefore, fraction [b] remains important. Beware: conducting variable selection on the union of the two explanatory data sets would make

fraction [b] very small or empty. If one wants to estimate how much of the variation of **Y** is explained jointly by the two explanatory data sets, it is important to carry out forward selection separately on the two sets of explanatory variables.

In this example, the two independent forward selections run above have retained the same variables as when the whole set had been submitted to forward selection (i.e. `alt`, `oxy`, `dbo`) *plus* variables `pen` and `nit`. The latter is strongly correlated to `alt` (r=−0.75). This means that in the RDA with the chemical variables, `nit` has explained some of the same structures as `alt` has explained in the physiography RDA. Removing `nit` from the chemical variables and running the partitioning again yields different results:

```
# 3. Variation partitioning without the 'nit' variable

envchem.pars2 <- envchem[,c(6,7)]
(spe.part2 <- varpart(spe.hel, envchem.pars2, envtopo.pars))
plot(spe.part2, digits=2)
```

This last partitioning has been run to demonstrate the effect of correlation among the explanatory variables of different sets. However, one generally applies variation partitioning to *assess* the magnitude of the various fractions, including the common ones. If the aim is to minimize the correlation among variables, other approaches are preferable (examination of the VIFs, global forward selection).

With this warning in mind, we can examine what the comparison between the two latter partitionings tells us. Interestingly enough, the overall amount of variation explained is about the same (0.595 instead of 0.590). The [b] fraction has dropped from 0.196 to 0.088 and the unique physiographical fraction (altitude + slope) has absorbed most of the difference, rising from 0.142 to 0.249. This does not mean that altitude is a better *causal* candidate than nitrates to explain fish communities. Comparison of the two analyses rather indicates that nitrate content is related to altitude *just as the fish communities are*, and that the interpretation of both variables must be done with caution since their causal link to the fish communities cannot be untangled. On the other hand, altitude is certainly related to other, unmeasured environmental variables that have an effect on the communities, making it a good proxy for them.

The comparisons above tell us that:

1. Forward selection provides a parsimonious solution without sacrificing real explanatory power: the R^2_{adj} of the three partitionings are approximately equal.

2. The common fractions, which are one of the reasons why partitioning is computed, must be interpreted with caution, even when the variables responsible for them are biologically legitimate.
3. Forward-selecting *all* explanatory variables before attributing the remaining ones to subsets is in contradiction with the aim of variation partitioning, except to help identify the magnitude of the effect of some variables responsible for the common fractions.
4. Forward selection and variation partitioning are powerful statistical tools, but they cannot replace sound ecological reasoning.

Final note about the [b] fraction. This fraction should *never* be mistaken for an interaction term in the analysis of variance sense. In a replicated two-way ANOVA, the interaction measures the influence of the levels of one factor on the effect of the other factor. It is most easily measured when the two factors are independent (uncorrelated, orthogonal) ... a case where the [b] fraction is equal to 0.

6.3.2.8 RDA as a Tool for Multivariate ANOVA

In its classical, parametric form, multivariate analysis of variance (MANOVA) has stringent conditions of application and restrictions (e.g. multivariate normality of each group of data, homogeneity of the variance–covariance matrices, number of response variables smaller than the number of objects minus the number of groups). It is practically never adapted to ecological data, despite its obvious interest for the analysis of the results of ecological experiments.

Fortunately, RDA offers an elegant alternative, while adding the versatility of the permutation tests and the triplot representation of results. The condition of homogeneity of the variance–covariance matrices still applies, however. It can be tested by the function **`betadisper()`** of package **`vegan`**. The trick is to use factor variables and their interactions as explanatory variables. In the example below, the factors are coded as orthogonal Helmert contrasts to allow testing the factors and interaction in a way that provides the correct F values. The interaction is represented by variables that are the products of the variables coding for the main factors. Properties (for a balanced design): (1) the sum of each coding variable is zero; (2) all variables are orthogonal (their scalar products are all zero); (3) the groups of variables coding for the main factors and their interaction are all orthogonal.

To illustrate this application, let us use a part of the Doubs data to construct a fictitious balanced two-way ANOVA design. We will use the first 27 sites, leaving the two last out.

Let us create a first factor representing altitude. This factor has three levels, one for each group of sites 1–10, 11–19 and 20–28 (remember that the empty site 8 has been removed from the data).

The second factor mimics the pH variable as closely as possible. In the real data, pH is more or less independent from altitude ($r = -0.05$), but no direct codification allows the creation of a three-level factor orthogonal to our first factor. Therefore, we simply create an artificial, three-level factor orthogonal to altitude and *approximately* representing pH. Be aware that we do this for illustration purposes only, and that such a manipulation would not be tolerable in the real world.

The end result is a balanced two-way crossed design with two factors of three levels each. After having tested for the homogeneity of variance–covariance matrices for the two factors, we test the two main factors and their interaction, each time using the appropriate terms as covariables. Using Helmert contrasts instead of factors allows an explicit control of all the terms of the model specification.

```
# Two-way MANOVA by RDA
# *********************

# Creation of a factor 'altitude' (3 levels, 9 sites each)
alt.fac <- gl(3,9)
# Creation of a factor mimicking 'pH'
pH.fac <-
 as.factor(c(1,2,3,2,3,1,3,2,1,2,1,3,3,2,1,1,2,3,2,1,2,3,2,1,1,3
 ,3))
# Are the factors balanced?
table(alt.fac, pH.fac)

# Creation of Helmert contrasts for the factors and their
interaction

alt.pH.helm <- model.matrix(~alt.fac * pH.fac,
contrasts=list(alt.fac="contr.helmert", pH.fac="contr.helmert"))
alt.pH.helm

 # Examine the table of contrasts. Which columns represent
 # alt.fac? pH.fac? The interaction?

# Check property 1 of Helmert contrasts: all variables sum to 0
apply(alt.pH.helm[,2:9],2,sum)
# Check property 2 of Helmert contrasts: variables are
# uncorrelated
cor(alt.pH.helm[,2:9])
```

```
# Verify multivariate homogeneity of within-group covariance
# matrices using the betadisper() function (vegan package)
# implementing Marti Anderson's testing method

spe.hel.d1 <- dist(spe.hel[1:27,])

# Factor "altitude"
(spe.hel.alt.MHV <- betadisper(spe.hel.d1, alt.fac))
anova(spe.hel.alt.MHV)
permutest(spe.hel.alt.MHV) # Permutational test

# Factor "pH"
(spe.hel.pH.MHV <- betadisper(spe.hel.d1, pH.fac))
anova(spe.hel.pH.MHV)
permutest(spe.hel.pH.MHV) # Permutational test

  # The within-group covariance matrices are homogeneous.
  # We can proceed.

# Test the interaction first. The factors alt and pH form the
# matrix of covariables

interaction.rda <- rda(spe.hel[1:27,], alt.pH.helm[,6:9],
alt.pH.helm[,2:5])
anova(interaction.rda, step=1000, perm.max=1000)

  # Is the interaction significant? A significant interaction
  # would preclude the analysis of the main factors since it
  # would indicate that the effect of a factor depends on the
  # level of the other factor.

# Test the main factor alt. The factor pH and the interaction
# form the matrix of covariables.

factor.alt.rda <- rda(spe.hel[1:27,], alt.pH.helm[,2:3],
 alt.pH.helm[,4:9])
anova(factor.alt.rda, step=1000, perm.max=1000, strata=pH.fac)

  # Is the factor alt significant?

# Test the main factor pH. The factor alt and the interaction
# form the matrix of covariables.

factor.pH.rda <- rda(spe.hel[1:27,], alt.pH.helm[,4:5],
 alt.pH.helm[,c(2:3, 6:9)])
anova(factor.pH.rda, step=1000, perm.max=1000, strata=alt.fac)
```

```
 # Is the factor pH significant?

# RDA and triplot for the significant factor alt

alt.rda.out <- rda(spe.hel[1:27,] ~ ., as.data.frame(alt.fac))
plot(alt.rda.out, scaling=1, display=c("sp", "wa", "cn"),
 main="Multivariate ANOVA, factor altitude - scaling 1 - wa
 scores")
spe.manova.sc <- scores(alt.rda.out, choices=1:2, scaling=1,
 display="sp")
arrows(0, 0, spe.manova.sc[, 1], spe.manova.sc[, 2], length=0,
  col="red")
```

Hints *To convince yourself that this procedure is the exact equivalent of an ANOVA, apply it to species 1 only, and then run a traditional ANOVA on species 1 with factors* `alt.fac` *and* `pH.fac`*, and compare the F values. The probabilities may differ slightly since they are permutational in RDA.*

In the permutational tests of each main effect, the argument `strata` *restricts the permutations within the levels of the other factor. This ensures that the proper H0 is produced by the permutation scheme.*

Examine the help files of functions **`model.matrix()`**, **`contrasts()`** *and* **`contr.helmert()`**.

If the design is unbalanced, this procedure can still be applied, but the contrasts are not orthogonal and therefore the tests of significance of the factors and interaction have reduced power to detect significant effects.

6.3.2.9 Distance-Based Redundancy Analysis

Ecologists have long needed methods for analysing community composition data in a multivariate framework. The need was particularly acute in ecological experiments designed to be analysed by multifactorial analysis of variance. We have seen that community composition data with large numbers of zeros must be transformed before they are used in MANOVA and other Euclidean-based models. The Legendre and Gallagher (2001) transformations (Sect. 3.5) are one way to solve that problem: transformed species data can be used in the ANOVA by RDA described in Sect. 6.3.2.8. These transformations cover only the chord, Hellinger, chi-square and Ochiai distance cases, however. Ecologists may want to compute RDA based on other dissimilarity measures that cannot be computed by a data transformation followed by the calculation of the Euclidean distance.

Legendre and Anderson (1999) proposed the method of distance-based redundancy analysis (db-RDA) to solve that problem. They showed that RDA could be used as a form of ANOVA which was applicable to community composition data if they were transformed in some appropriate way, which went through the calculation of a dissimilarity matrix of the user's choice. This approach remains fully valid and useful for all dissimilarity measures that cannot be obtained by a data transformation followed by the calculation of the Euclidean distance. Among the ones devoted to communities of living organisms, let us mention most measures for binary data (e.g. Jaccard $\left(\sqrt{1-S_7}\right)$, Sørensen $\left(\sqrt{1-S_8}\right)$), and quantitative distance measures like Bray–Curtis (D_{14}), asymmetric Gower $\left(\sqrt{1-S_{19}}\right)$, Whittaker ($D_9$) and Canberra ($D_{10}$). Dissimilarities intended for other types of data, e.g. symmetric Gower $\left(\sqrt{1-S_{15}}\right)$, Estabrook–Rogers $\left(\sqrt{1-S_{16}}\right)$, and the generalized Mahalanobis distance for groups of observations, can also be used in canonical ordination through db-RDA. Examples of papers involving db-RDA are Anderson (1999), Geffen et al. (2004) and Lear et al. (2008). The method goes as follows:

- Compute a Q-mode dissimilarity matrix for the response data.
- Compute a principal coordinate analysis (PCoA) of the dissimilarity matrix, correcting for negative eigenvalues if necessary. Keep all principal coordinates in a file; they express all the variance of the data as seen through the dissimilarity measure.
- Run and test an RDA of the principal coordinates created above (acting as response data) constrained by the explanatory variables available in the study. The explanatory variables may represent the factors of a manipulative or mensurative experiment.

These steps are quite simple; they can be run one by one in R with a few lines of code only, but a shortcut is available in **vegan**, as will be shown below. For the sake of example, we will compute a Bray–Curtis dissimilarity matrix of the fish data before submitting it to the same ANOVA-like RDA as above.

```
# Distance-based redundancy analysis (db-RDA)
# ******************************************

# 1. Explicit steps
spe.bray <- vegdist(spe[1:27,], "bray")
spe.pcoa <- cmdscale(spe.bray, k=nrow(spe[1:27,])-1, eig=TRUE,
 add=TRUE)
spe.scores <- spe.pcoa$points

# Test of the interaction. Factors alt and pH, Helmert-coded,
# form the matrix of covariables
```

```
interact.dbrda <- rda(spe.scores[1:27,], alt.pH.helm[,6:9],
 alt.pH.helm[,2:5])
anova(interact.dbrda, step=1000, perm.max=1000)

 # Is the interaction significant?
 # If not, you can proceed and test the main factors (not shown)

# 2. Direct way using vegan's function capscale. Runs with model
#    interface only.
#    Response matrix can be raw data:

interact.dbrda2 <- capscale(spe[1:27,] ~ alt.fac*pH.fac +
Condition(alt.fac+pH.fac), distance="bray", add=TRUE)
anova(interact.dbrda2, step=1000, perm.max=1000)

# Or response matrix can be a dissimilarity matrix:
interact.dbrda3 <- capscale(spe.bray ~ alt.fac*pH.fac
  + Condition(alt.fac+pH.fac), add=TRUE)
anova(interact.dbrda3, step=1000, perm.max=1000)
```

Hints *In the* ***cmdscale()*** *and* ***capscale()*** *functions, the argument* `add=TRUE` *adds a constant to the distances to avoid negative eigenvalues. This is the Cailliez correction.*

When using raw response data, the association coefficient is determined by the argument `distance`.

6.3.2.10 Nonlinear Relationships in RDA

One last point is worth mentioning. RDA carries out a multivariate linear regression analysis followed by a PCA of the fitted values. Consequently, other techniques used in multiple regression can be used in RDA as well. Consider the fact that all RDA models presented above used only first-degree explanatory variables. However, it is frequent that *raw* species responses are in fact unimodal, with a species showing an ecological optimum and some tolerance to the variation of a given environmental constraint. A strictly linear model would suffer from lack-of-fit in such a case. Plotting all possible pairs of response versus explanatory variables to detect such non-linearities would be too cumbersome. An interesting shortcut to identify and model unimodal responses is to provide second-degree explanatory variables along with the first degree terms (i.e. provide the terms for a quadratic model) and run forward selection. This procedure will retain the relevant variables, be they of the first or second degree. Of course, the interpretation of such results is

more complex, so that it should be applied only when one has serious reasons to suspect non-linear relationships. Third-order explanatory variables may even be useful in the case of unimodal but highly skewed response variable distributions. Raw polynomial terms are highly correlated, so it is preferable to use orthogonal polynomials, which can be computed by function **poly()** of the package **stats**.

In the Doubs data, there are several species that are found mostly in the middle section of the river. Their relationship with the variable "distance from the source" (das) is therefore unimodal: absence first, then presence, then absence again. Such a simple case could be a good candidate to experiment with a second-degree variable. When interpreting the result, note that species having their optimum around mid-river will point to the *opposite* direction from the das-squared variable das2, because the quadratic term of a unimodal model is negative. Species with arrows pointing in the same direction as the das2 variable may be more present at both ends of the river than in the middle. Remember also that this analysis is done on *untransformed* response variables.

```
# RDA with a second degree explanatory variable
# ********************************************

# Create a matrix of das and its orthogonal second degree term
# using poly()
das.df <- poly(das, 2)
colnames(das.df) <- c("das", "das2")
# Verify if both variables are significant
forward.sel(spe, das.df)

# RDA and test
spe.das.rda <- rda(spe ~ ., as.data.frame(das.df))
anova(spe.das.rda)

# Triplot using lc (model) site scores and scaling 2
plot(spe.das.rda, scaling=2, display=c("sp","lc","cn"),
 main="Triplot RDA spe ~ das+das2 - scaling 2 - lc scores")
spe6.sc <- scores(spe.das.rda, choices=1:2, scaling=2,
 display="sp")
arrows(0, 0, spe6.sc[,1], spe6.sc[,2], length=0, lty=1,
 col="red")
```

Hint *If one computes raw second-degree variables by hand (i.e. by squaring first-degree variables) or by applying argument* raw=TRUE *to function* **poly()**, *it is better to* ***centre*** *the first-degree variables before computing the second-degree terms, otherwise the latter will be strongly linearly related to the former. This is not necessary when using* **poly()**.

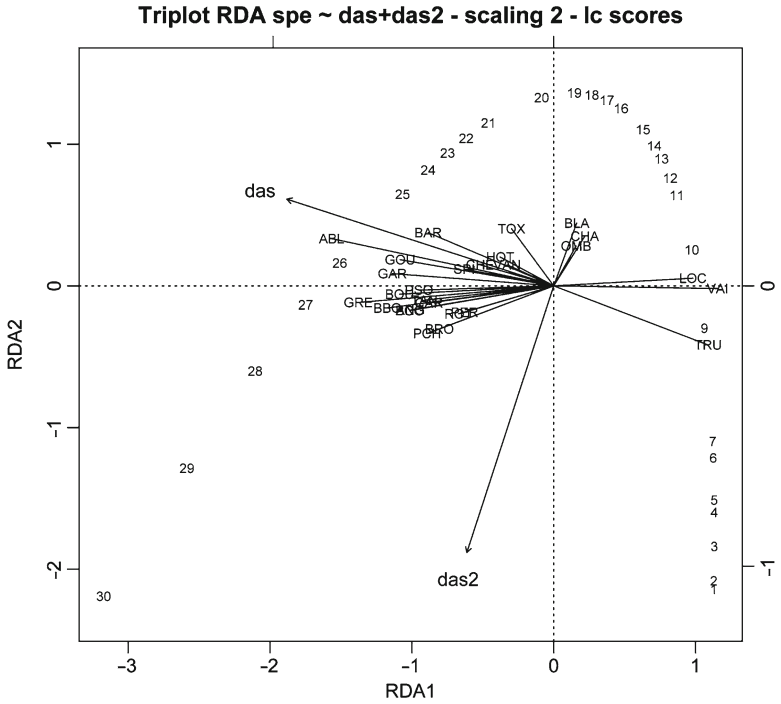

Fig. 6.6 Scaling 2 triplot of the untransformed fish species explained by an orthogonal second degree polynomial of the variable "distance from the source" (`das`)

We drew a scaling 2 triplot because we were interested primarily in the relationships among the species (Fig. 6.6). At first glance, this triplot looks like some mistake has been made, but in fact it displays exactly what has been asked for. The surprising feature is, of course, the curved distribution of the sites in the plot. Since we modelled the data by means of a second-degree function (i.e. a parabolic function) of the distance from the source, the modelled data (option "`lc`") form therefore a parabola. If you want a triplot showing the sites in a configuration closer to the data, replace "`lc`" by "`wa`" in the plot function above.

To illustrate the interpretation of the first-order `das` and the second-order `das2` variables, let us take some fish species as examples. We map them along the river to make the comparison easier. The four examples are the brown trout (`TRU`), the grayling (`OMB`), the bleak (`ABL`) and the tench (`TAN`). The code is the same as in Chap. 2 (Fig. 6.7).

```
# Maps of four fish species
# ************************
par(mfrow=c(2,2))

plot(spa$x, spa$y, asp=1, col="brown", cex=spe$TRU,
  xlab="x (km)", ylab="y (km)", main="Brown trout")
lines(spa$x, spa$y, col="light blue")

plot(spa$x, spa$y, asp=1, col="brown", cex=spe$OMB,
  xlab="x (km)", ylab="y (km)", main="Grayling")
lines(spa$x, spa$y, col="light blue")

plot(spa$x, spa$y, asp=1, col="brown", cex=spe$ABL,
  xlab="x (km)", ylab="y (km)", main="Bleak")
lines(spa$x, spa$y, col="light blue")

plot(spa$x, spa$y, asp=1, col="brown", cex=spe$TAN,
  xlab="x (km)", ylab="y (km)", main="Tench")
lines(spa$x, spa$y, col="light blue")
```

Comparing the ordination triplot with these four maps shows how to interpret the fish vectors in combination with the two variables `das` and `das2`. Among all species, the brown trout `TRU` is most strongly linked to the upper half of the river; its vector is opposed to `das` and orthogonal to (i.e. independent of) `das2`; the grayling (`OMB`) is characteristic of the middle part of the river. Note that its vector on the triplot is *opposed* to that of variable `das2`. The bleak (`ABL`) is abundant in the lower half of the river, as confirmed by its vector pointing in the direction of `das`, orthogonal to `das2` and directly opposite to TRU. Finally, the tench (`TAN`) is present in three different zones along the river, which results in vector pointing halfway between the `das` and `das2` vectors.

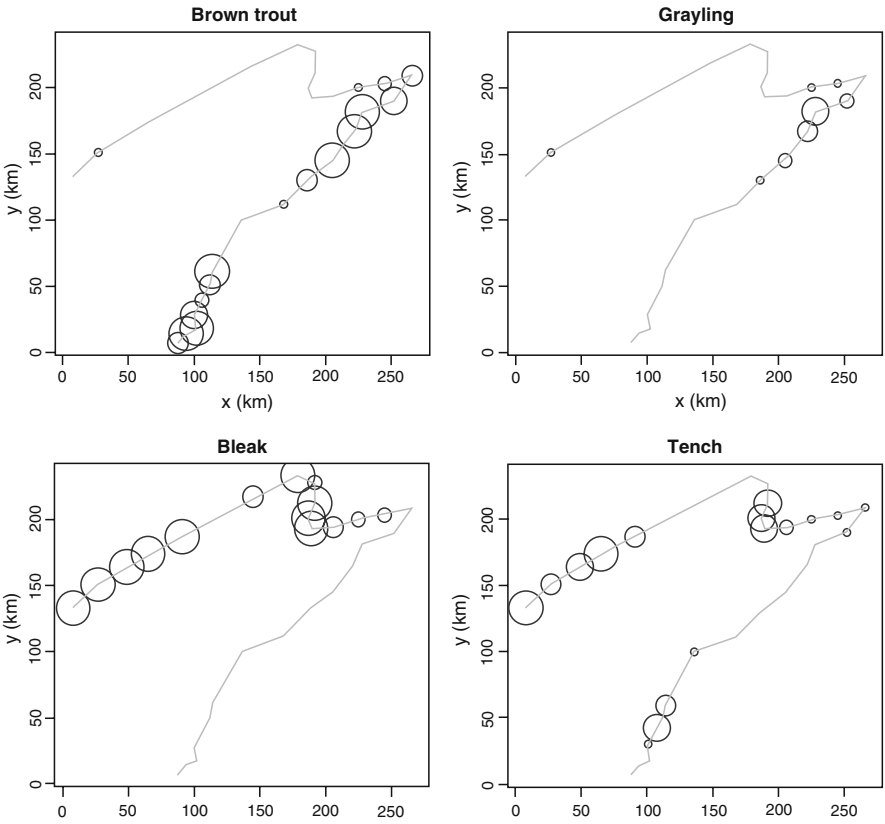

Fig. 6.7 Bubble plots of the abundances of four fish species, to explain the interpretation of the second degree RDA presented above

Of course, since we have other, more explicit environmental variables at our disposal, this exercise may seem unnecessary. But it shows that in other cases, adding a second-degree term may indeed add explanatory power to the model and improve its fit: the forward selection result confirmed that `das2` added a significant contribution in this example, the R^2_{adj} being raised by 6.3%.

The use of higher-degree explanatory variables is typical in the frame of spatial analysis, where the explanatory variables are spatial coordinates and the higher-degree variables form trend-surface analysis models (see also Chap. 7). This does not, however, preclude their use with other types of explanatory variables.

6.3.3 A Hand-Written RDA Function

The code below is another exercise in matrix algebra; just follow the equations.

The Code It Yourself corner #4

To be able to code an RDA from scratch, we will refer to the algebra provided by Legendre and Legendre (1998), Section 11.1. The steps are the following (RDA on a covariance matrix):

1 - Compute the multivariate linear regression of the centred data matrix on the standardized matrix of explanatory variables. Obtain the matrix of fitted values.

2 – Compute PCA on the matrix of fitted values.

3 - Compute the two types of site scores.

4 - Output of the results.

The equations used are referred to in the comments, using Legendre and Legendre's numbering.

The code below concentrates on the constrained part of the RDA. It is intended for readers interested in the mathematical procedure of RDA and is not optimized for performance.

```
myRDA <- function(Y,X) {

# 1. Preparation of the data
# **************************
   Y.mat <- as.matrix(Y)
   Yc <- scale(Y.mat, scale=FALSE)
   X.mat <- as.matrix(X)
   Xcr  <- scale(X.mat)

# 2. Computation of the multivariate linear regression
# ****************************************************

# Matrix of regression coefficients (eq. 11.4)
   B <- solve(t(Xcr) %*% Xcr) %*% t(Xcr) %*% Yc

# Matrix of fitted values (eq. 11.5)
   Yhat <- Xcr %*% B

# Matrix of residuals
   Yres <- Yc - Yhat
```

```
# Dimensions
   n <- nrow(Y)
   p <- ncol(Y)
   m <- ncol(X)

# 3. PCA on fitted values
# ***********************

# Covariance matrix (eq. 11.7)
   S <- cov(Yhat)

# Eigenvalue decomposition
   eigenS <- eigen(S)

# How many canonical axes?
   kc <- length(which(eigenS$values > 0.00000001))

# Eigenvalues of canonical axes
   ev <- eigenS$values[1:kc]
# Total variance (inertia) of the centred matrix Yc
   trace = sum(diag(cov(Yc)))

# Orthonormal eigenvectors (contributions of response
# variables, scaling 1)
   U <- eigenS$vectors[,1:kc]
   row.names(U) <- colnames(Y)

# Site scores (vegan's 'wa' scores, scaling 1; eq. 11.12)
   F <- Yc %*% U
   row.names(F) <- row.names(Y)

# Site constraints (vegan's 'lc' scores, scaling 1; eq. 11.13)
   Z <- Yhat %*% U
   row.names(Z) <- row.names(Y)

# Canonical coefficients (eq. 11.14)
   CC <- B %*% U
   row.names(CC) <- colnames(X)

# Explanatory variables
# Species-environment correlations
   corXZ <- cor(X,Z)

# Diagonal matrix of weights
   D <- diag(sqrt(ev/trace))

# Biplot scores of explanatory variables
   coordX <- corXZ %*% D    # Scaling 1
   coordX2 <- corXZ         # Scaling 2
   row.names(coordX) <- colnames(X)
```

```
   row.names(coordX2) <- colnames(X)

# Scaling to sqrt of the relative eigenvalue (for scaling 2)
   U2 <- U %*% diag(sqrt(ev))
   row.names(U2) <- colnames(Y)
   F2 <- F %*% diag(1/sqrt(ev))
   row.names(F2) <- row.names(Y)
   Z2 <- Z %*% diag(1/sqrt(ev))
   row.names(Z2) <- row.names(Y)

# Unadjusted R2
   R2 <- sum(ev/trace)

   # Adjusted R2
   R2adj <- 1-((n-1)/(n-m-1))*(1-R2)

# 4. PCA on residuals
# ******************
# ... write your own code as in Chapter 5. It could begin with:
#     eigenSres <- eigen(cov(Yres))
#     evr <- eigenSres$values

# 5. Output
# *********
   result <- list(trace, R2, R2a,ev,CC,U,F,Z,coordX,
       U2,F2,Z2,coordX2)
   names(result) <- c("Total variance", "R2", "R2a","Can ev",
       "Can coeff", "Species sc1", "wa sc1", "lc sc1",
       "Biplot sc1", "Species sc2", "wa sc2", "lc sc2",
       "Biplot sc2")
   result
}
```

Apply the function to the Doubs fish and environmental data

```
doubs.myRDA <- myRDA(spe.hel env)
summary(doubs.myRDA)
```

6.4 Canonical Correspondence Analysis

6.4.1 Introduction

The canonical counterpart of CA, canonical correspondence analysis, has been acclaimed by ecologists ever since its introduction (ter Braak, 1986, 1987, 1988). It shares many characteristics with RDA, so that a detailed description is not necessary here. Basically, it is a weighted form of RDA applied to the same matrix $\overline{Q}$ of contributions to the χ^2 statistic as used in CA (Legendre and Legendre 1998, Section 11.2). CCA shares the basic properties of CA, combined with those of a constrained ordination. It preserves the χ^2 *distance* among sites, and species are represented as points in the triplots. ter Braak (1986) has shown that, provided that some conditions are fulfilled,[3] CCA is a good approximation of a multivariate Gaussian regression. One particularly attractive feature of a CCA triplot is that species are ordered along the canonical axes following their ecological optima. This allows a relatively easy ecological interpretation of species assemblages. Also, species scores can be used as synthetic descriptors in a clustering procedure (for instance *k*-means partitioning) to produce a typology of the species in assemblages.

CCA does have some drawbacks, however, related to the mathematical properties of the χ^2 distance. Legendre and Gallagher (2001) state that "a difference between abundance values for a common species contributes less to the distance than the same difference for a rare species, so that rare species may have an unduly large influence on the analysis". Despite its widespread use, "the χ^2 distance is not unanimously accepted among ecologists; using simulations, Faith et al. (1987) concluded that it was one of the worst distances for community composition data" (Legendre and Gallagher 2001). Its use should be limited to situations where rare species are well sampled and are seen as potential indicators of particular characteristics of an ecosystem; the alternative is to eliminate rare species from the data table before CCA. These problems, among other points, have led to the development of the species pretransformations to open these data to the realm of RDA, ANOVA and other linear methods. Furthermore, the proportion of total inertia represented by explained inertia (inertia is the measure of explained variation of the data in CCA), which can be interpreted as an R^2, is also biased, but no simple method exists for its adjustment. Ezekiel's adjustment cannot be used. A rather

[3] Two important conditions are that the species must have been sampled along their whole ecological range and that they display unimodal responses toward their main ecological constraints. These conditions are difficult to test formally, but graphs of species abundances in sites arranged along their scores on the first few ordination axes may help visualize their distributions along the main ecological gradients.

cumbersome bootstrap procedure can be used for its estimation, for which no **R** code exists as yet (Peres-Neto et al. 2006). This precludes a correct estimation of the proportion of variation explained by CCA in the case of a single analysis, and unduly inflates the type I error and the estimation of explained variation in forward selection since Blanchet et al. (2008a)'s double stopping criterion cannot be applied. Since one cannot take the number of explanatory variables into account in computing an adjusted R^2, the estimated relative amounts of variation in variation partitioning are also distorted.

Despite these shortcomings, CCA is still widely used and deserves an illustration.

6.4.2 CCA of the Doubs Data

6.4.2.1 CCA Using `vegan`

Let us run a CCA of the Doubs data using the formula interface. The species data are the raw, untransformed abundances (do *not* use the Hellinger-transformed data, which are meant to be used with RDA; the preserved distance would no longer be the χ^2 distance and the results could not be interpreted).

```
# CANONICAL CORRESPONDENCE ANALYSIS (CCA)
# ***************************************

# CCA of the raw fish species data, constrained by all the
# environmental variables in env2
spe.cca <- cca(spe ~ ., env2)
spe.cca
summary(spe.cca)  # Scaling 2 (default)
```

The differences with an RDA output are the following:

- The variation is now expressed as *Mean squared contingency coefficient*. It is biased but cannot easily be adjusted.
- The maximum number of canonical axes in CCA is $\min[(p-1), m, n-1]$. The minimum number of residual axes is $\min[(p-1), n-1]$. In our example, these numbers are the same as in RDA.
- The species scores are represented by *points* in the triplot.
- Site scores are weighted *averages* (instead of weighted sums) of species scores.

6.4.2.2 CCA Triplot

The code to produce CCA triplots is similar to the one used for RDA, except that the response variables (species) are represented by points and thus arrows are not necessary for them.

```
# CCA triplots (using lc site scores)
# ***********************************

# Scaling 1: species scores scaled to relative eigenvalues,
# sites are weighted averages of the species
plot(spe.cca, scaling=1, display=c("sp","lc","cn"),
main="Triplot CCA spe ~ env2 - scaling 1")

# Default scaling 2: site scores scaled to relative eigenvalues,
# species are weighted averages of the sites
plot(spe.cca, display=c("sp","lc","cn"), main="Triplot CCA spe ~
env2 - scaling 2")
```

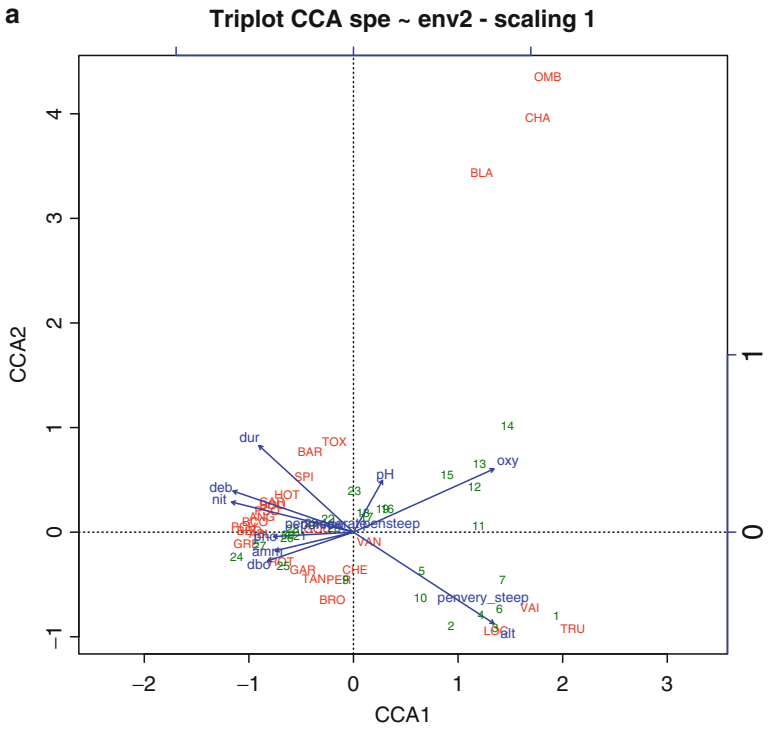

Fig. 6.8 (**a**) CCA triplot of the Doubs fish species constrained by all environmental variables except `das`. Scaling 1

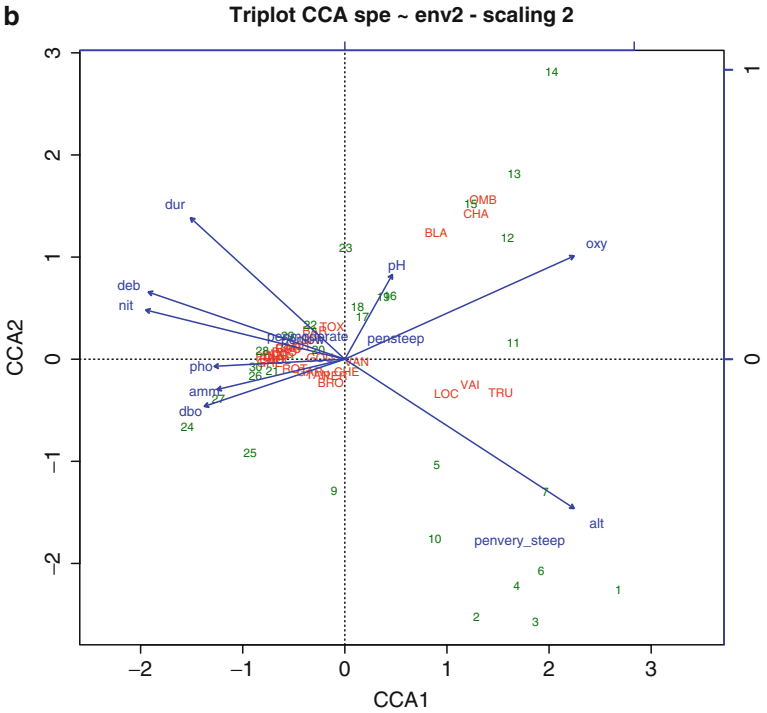

Fig. 6.8 (continued) (**b**) CCA triplot of the Doubs fish species constrained by all environmental variables except `das`. Scaling 2

In CCA as in RDA, the introduction of a third entity (explanatory variables) calls for additional interpretation rules for the triplots. Here are the essential ones:

- *Scaling 1* – (1) Projecting an object at right angle on a **quantitative explanatory variable** approximates the position of the object along that variable. (2) An object found near the point representing the centroid of a **qualitative explanatory variable** is more likely to possess state "1" for that variable. (3) Distances among **centroids** of qualitative explanatory variables, and between **centroids** and individual objects, approximate χ^2 distances.
- *Scaling 2* – (1) The optimum of a species along a **quantitative environmental variable** can be obtained by projecting the species at right angle on the variable. (2) A species found near the **centroid of a qualitative environmental variable** is likely to be found frequently (or in larger abundances) in the sites possessing state "1" for that variable. (3) *Distances among **centroids**, and between centroids and individual objects, do not approximate χ^2 distances.*

The scaling 1 triplot focuses on the distance relationships among sites, but the presence of species with extreme scores renders the plot difficult to interpret beyond trivialities. Therefore, it may be useful to redraw it without the species (Fig. 6.9a):

```
# CCA scaling 1 biplot without species (using lc site scores)
# ***********************************************************

plot(spe.cca, scaling=1, display=c("lc","cn"), main="Biplot CCA
  spe ~ env2 - scaling 1")


# CCA scaling 2 biplot without sites
# **********************************

plot(spe.cca, scaling=2, display=c("sp", "cn"), main="Biplot CCA
  spe ~ env2 - scaling 2")
```

Here the response of the fish communities to their environmental constraints is more apparent. One can see two well-defined groups of sites, one linked to high altitude and very steep slope (sites 1–7 and 10) and another with the highest oxygen contents (sites 11–15). The remaining sites are distributed among various conditions towards more eutrophic waters. Remember that this is a constrained ordination of the *fish* community data, not a PCA of the site environmental variables. The triplot therefore displays how the fish community is organized with respect to the environmental constraints.

The scaling 2 triplot (Fig. 6.8b) shows two groups of species: `OMB`, `CHA` and `BLA` linked to high oxygen concentrations; `TRU`, `VAI` and `LOC` linked to high altitude and very steep slope. To help untangle the other species, a biplot without the sites may be useful. You could plot it as we did above for the scaling 1 biplot, but this time leaving out the site (`"lc"`) scores and using species (`"sp"`) scores (Fig. 6.9b).

This closer look shows that `ROT`, `CAR`, `TAN`, `BRO`, `GOU` and `PER` are linked to high ammonium and phosphate concentrations, as well as high biological oxygen demand; most other species are linked to high nitrate concentrations, moderate to low slopes and high discharge.

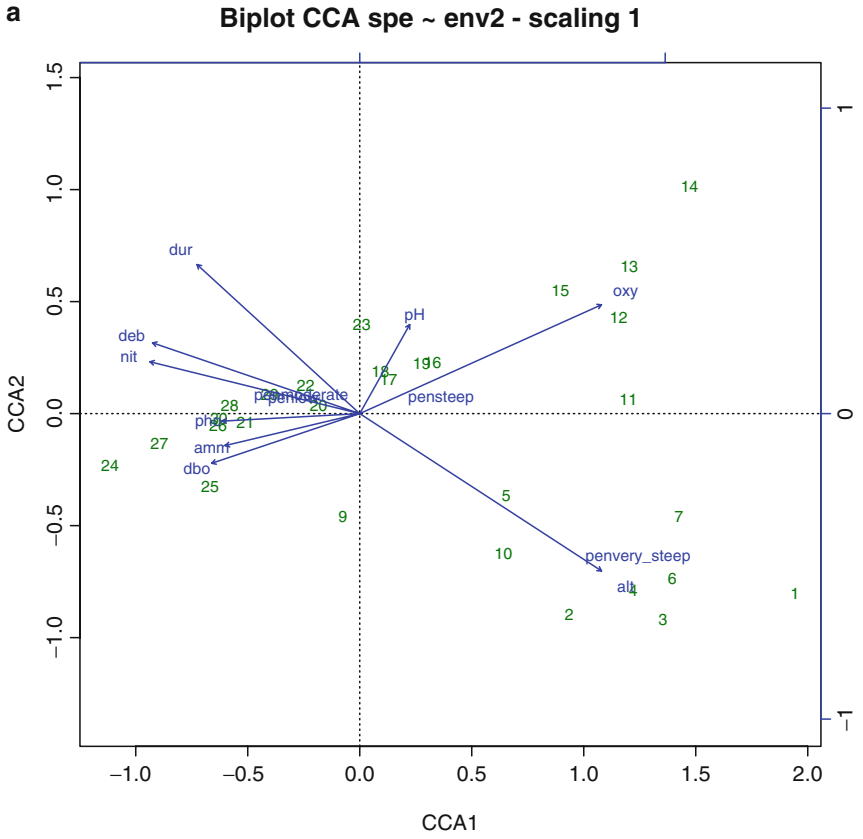

Fig. 6.9 (**a**) Biplot of CCA, scaling 1 with fitted site scores

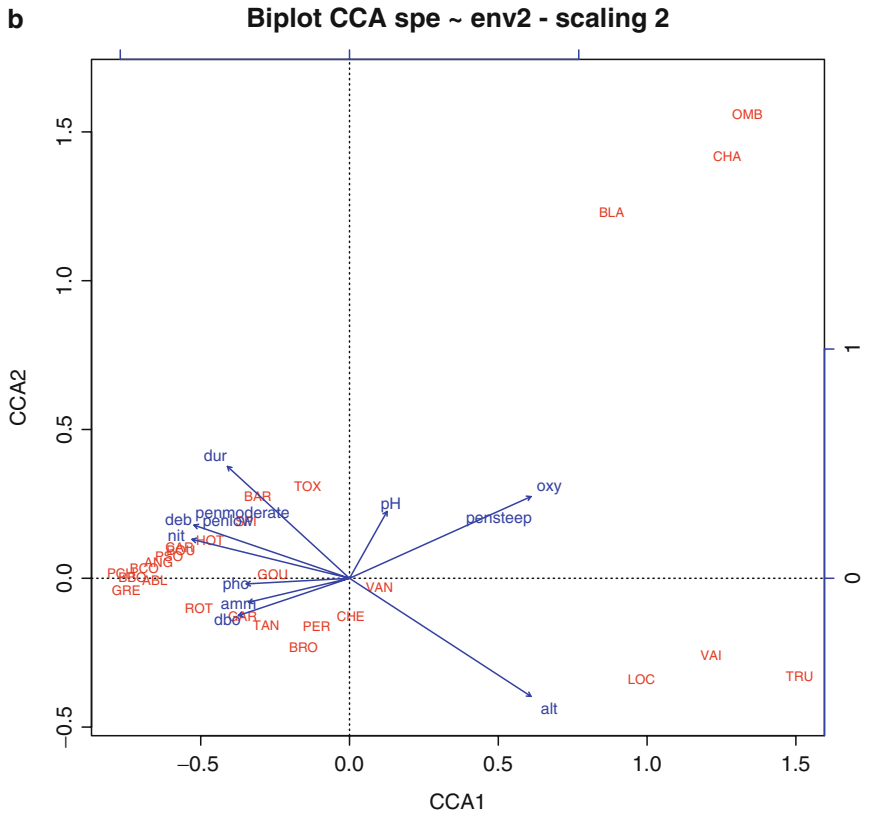

Fig. 6.9 (continued) (**b**) Biplot of CCA, scaling 2 with species scores

6.4.2.3 Permutation Tests in CCA, Forward Selection and Parsimonious CCA

CCA results can be tested for significance by permutations, in the same way as RDA.

```
# Permutation tests of CCA results
# ********************************

# Permutation test of the overall analysis
anova(spe.cca, step=1000)

# Permutation test of each axis
anova(spe.cca, by="axis", step=1000)
```

The RDA presented in Sect. 6.3.2.2, although globally significant, was not parsimonious. Therefore, we computed a forward selection of explanatory variables (Sect. 6.3.2.6). Let us do the same thing here with function **`ordistep()`**, since **`forward.sel()`** only computes RDA.

```
# CCA-based forward selection using vegan's ordistep()
# ****************************************************
# This function allows the use of factors like 'pen' in env2

cca.step.forward <- ordistep(cca(spe ~ 1, data=env2),
  scope=formula(spe.cca), direction="forward", pstep=1000)
```

The result is the same as the most parsimonious one based on RDA. Therefore, we can compute a parsimonious CCA on the basis of the same three explanatory variables: altitude, oxygen concentration and biological oxygen demand.

```
# Parsimonious CCA using alt, oxy and dbo
# ***************************************

(spe.cca.pars <- cca(spe ~ alt + oxy + dbo, data=env2))
anova.cca(spe.cca.pars, step=1000)
anova.cca(spe.cca.pars, step=1000, by="axis")

vif.cca(spe.cca)
vif.cca(spe.cca.pars)
```

Hint *Although the explanatory variables are the same, the VIFs differ from those of RDA because CCA is a* ***weighted*** *regression procedure where the (raw) species data have undergone a double standardization (per rows and columns).*

As in RDA, parsimony has paid off. The unadjusted explained inertia is now 56.3%; it was 72.5% with all explanatory variables. This drop (which would certainly be smaller if these values were adjusted) is compensated by a clearer model with three significant canonical axes. The VIFs of the three remaining variables are around 3, which is good.

6.4.2.4 Three-Dimensional Interactive Plots

Instead of plotting these parsimonious results as we did before, let us explore a **vegan** function that can be very useful either for a researcher looking for a new perspective on his or her results, or for a teacher: a 3D interactive plot. We see several options to reveal or combine results in different ways. These 3D plots are run under the `rgl` package.

```
# Three-dimensional interactive ordination plots
# **********************************************

# Plot of the sites only (wa scores)
# **********************************
ordirgl(spe.cca.pars, type="t", scaling=1)
  # Using the mouse, enlarge the plot by dragging its lower
  # right-hand corner. Then move the plot around by left-clicking
  # on any point in the plot with the left button. Use the scroll
  # wheel to zoom in and out.

# Connect weighted average scores to linear combination scores
orglspider(spe.cca.pars, scaling=1, col="purple")
  # The purple connections show how well the CCA model fits the
  # data. The shorter the connections, the better the fit.

# Plot the sites (wa scores) with a clustering result
# ***************************************************
# Colour sites according to cluster membership
gr <- cutree(hclust(vegdist(spe.hel, "euc"), "ward"), 4)
ordirgl(spe.cca.pars, type="t", scaling=1, ax.col="black",
 col=gr+1)
# Connect sites to cluster centroids
orglspider(spe.cca.pars, gr, scaling=1)
  # The sites are nicely clustered along their major ecological
  # gradients. Remember that this is an analysis of the fish
  # community data.

# Complete CCA 3D triplot
# ***********************
ordirgl(spe.cca.pars, type="t", scaling=2)
```

```
orgltext(spe.cca.pars, display="species", type="t", scaling=2,
  col="cyan")

# Plot species groups (Jaccard similarity, useable in R mode)
# ************************************************************
gs <- cutree(hclust(vegdist(t(spe), method="jaccard"), "ward"),
  4)
ordirgl(spe.cca.pars, display="species", type="t", col=gs+1)

rgl.quit()    # shutdown rgl device system
```

Hint *Three-dimensional plots have many options. Type* **`?ordirgl`** *to explore some of them. It is also possible to draw 3D plots of RDA results, but there is no simple means to draw arrows for the response variables.*

6.5 Linear Discriminant Analysis

6.5.1 Introduction

LDA differs from RDA and CCA in that the response variable is a grouping of the sites. This grouping may have been obtained by clustering the sites on the basis of a data set, or it may represent an ecological hypothesis. LDA tries to determine to what extent an *independent* set of quantitative variables can explain this grouping. We insist that the site typology must have been obtained independently from the explanatory variables used in the LDA; otherwise, the procedure would be circular and the tests would be invalid.

LDA computes *discriminant functions* from standardized descriptors. These coefficients quantify the relative contributions of the (standardized) explanatory variables to the discrimination of objects. On the other hand, *identification functions* can be computed from the original (not standardized) descriptors and can be used to find the group to which a new object should be attributed. The example below shows both operations (discrimination and identification).

To perform LDA, one must ensure that the within-group covariance matrices of the explanatory variables are homogeneous, a condition that is frequently violated with ecological data.

6.5.2 Discriminant Analysis Using `lda()`

`lda()` is a function of package **`MASS`**. As a simple example, we can use the four-group classification of sites based on the fish species (`gr` in the 3D plots above), and try to explain this classification using the three environmental variables that have been selected in Sect. 6.3.2.6.

When interpreting the result, we refer to the equations presented in Legendre and Legendre (1998, Section 11.5).

```
# LINEAR DISCRIMINANT ANALYSIS (LDA)
# **********************************

env.pars2 <- as.matrix(env[,c(1,9,10)])

# Verify multivariate homogeneity of within-group covariance
# matrices using the betadisper() function (vegan package)
# implementing Marti Anderson's testing method

env.pars2.d1 <- dist(env.pars2)
(env.MHV <- betadisper(env.pars2.d1, gr))
anova(env.MHV)
permutest(env.MHV)  # Permutational test

  # The within-group covariance matrices are NOT homogeneous.
  # Let us try a log transformation of variables alt and dbo

env.pars3 <- cbind(log(env$alt), env$oxy, log(env$dbo))
colnames(env.pars3) <- c("alt.ln", "oxy", "dbo.ln")
row.names(env.pars3) <- row.names(env)
env.pars3.d1 <- dist(env.pars3)
(env.MHV2 <- betadisper(env.pars3.d1, gr))
permutest(env.MHV2)

  # This time the within-group covariance matrices are
  # homogeneous. We can proceed.

# Computation of LDA (discrimination)

env.pars3.df <- as.data.frame(env.pars3)
(spe.lda <- lda(gr ~ alt.ln + oxy + dbo.ln, data=env.pars3.df))

# The result object contains the information necessary to
# interpret the LDA
summary(spe.lda)
```

```
# Display the group means for the 3 variables
spe.lda$means

# Compute the normalized eigenvectors (matrix C, eq. 11.33)
# which are the standardized discriminant function coefficients
(Cs <- spe.lda$scaling)

# Compute the canonical eigenvalues
spe.lda$svd^2

# Position the objects in the space of the canonical variates
(Fp <- predict(spe.lda)$x)
# alternative way: Fp <- scale(env.pars3.df, center=TRUE,
scale=FALSE) %*% C

# Classification of the objects
(spe.class <- predict(spe.lda)$class)

# Posterior probabilities of the objects to belong to the groups
(spe.post <- predict(spe.lda)$posterior)

# Table of prior versus predicted classifications
(spe.table <- table(gr, spe.class))

# Proportion of correct classification
diag(prop.table(spe.table, 1))

# Plot the objects in the space of the canonical variates
# with colours according to their classification
plot(F[,1], F[,2], type="n")
text(F[,1], F[,2], row.names(env),
col=c(as.numeric(spe.class)+1))
abline(v=0, lty="dotted")
abline(h=0, lty="dotted")
# Draw 95% ellipses around the groups
for(i in 1:length(levels(as.factor(gr)))){
   cov <- cov(Fp[gr==i,])
   centre <- apply(Fp[gr==i,], 2, mean)
   lines(ellipse(cov, centre=centre, level=0.95))
}

# Classification of new object (identification)
# A new object is created with ln(alt)=6.8, oxygen=90 and
ln(dbo)=3.2

newo <- c(6.8,90,3.2)
```

```
newo <- as.data.frame(t(newo)) # Must be a row
colnames(newo) <- colnames(env.pars3)
(predict.new <- predict(spe.lda, newdata=newo))
```

The new object has been classified in group 1. This calculation could have been done in the same way for a whole table of new observations. Now you could examine the profiles of the fish species of group 1. What you have actually done is to forecast that a site with the environmental values found in vector "new" should contain this type of fish community.

```
# LDA with jackknife-based classification (i.e., leave-one-out
# cross-validation)

spe.lda.jac <- lda(gr ~ alt.ln + oxy + dbo.ln,
data=env.pars3.df, CV=TRUE)
summary(spe.lda.jac)

# Numbers and proportions of correct classification
spe.jac.class <- spe.lda.jac$class
spe.jac.table <- table(gr, spe.jac.class)
diag(prop.table(spe.jac.table, 1))
```

The classification success in `spe.jac.table` seems not as good as the result in `spe.table`. Remember, however, that `spe.table` shows an *a posteriori* classification of the objects that have been used in the computations. It is too optimistic. By comparison, cross-validation results are obtained by computing the "lda" and classification of each object, in turn, with that object taken out of the "lda" calculation. It is more realistic.

6.6 Other Asymmetrical Analyses

Not all possible forms of multivariate analysis have been developed above. There are several additional methods that may prove useful in some applications. Most of them are derived from RDA. Among them, let us mention Principal response curves (PRC; Van den Brink and ter Braak 1998, 1999), the asymmetric form of co-correspondence analysis (ter Braak and Schaffers 2004) and a method to test the space–time interaction in surveys through space and time without replication at the level of individual sampling units (Legendre et al. 2010).

PRC are designed to analyse treatment effects over time in terms of differences between control and treatment; they provide a clear graphical illustration of

treatment effects at the community as well as the species level. They are available through the function **prc()** of package **vegan**.

Co-correspondence analysis is based on correspondence analysis and is devoted to the simultaneous ordination of two communities sampled at the same sites. Its asymmetric form allows one to predict a community on the basis of the other. It can be computed with an **R** package called **cocorresp**.

Space–time interaction testing is based on RDA and a spatial filtering technique (MEM) described in Chap. 7. An **R** package called **STI** is available through an ESA archive referenced in the paper.

6.7 Symmetrical Analysis of Two (or More) Data Sets

"Symmetrical analysis" means that the two matrices involved in the analysis play the same role; there is no "dependent" or "explanatory" matrix. The choice between symmetrical and asymmetrical ordination methods is akin to the choice between correlation and model I regression analysis. The former is more descriptive or exploratory, and also appropriate when no unidirectional causal hypothesis is imbedded in the model, while the latter is more inferential, i.e. oriented at explaining the variation of *response* variables by means of a (hopefully parsimonious) linear combination of *explanatory* variables. The two approaches are therefore complementary; they fulfil different research aims and should not be opposed as competitors on the same terrain.

Three symmetrical methods are presented here because of their interest in ecology: canonical correlation analysis (CCorA), co-inertia analysis (CoIA) and multiple factor analysis (MFA). Another method, the symmetric form of co-correspondence analysis, is devoted to the simultaneous ordination of two communities. It can also be computed with the package **cocorresp**.

6.8 Canonical Correlation Analysis

6.8.1 Introduction

CCorA is computed on two data tables. The aim of the method is to represent the observations along canonical axes that maximize the correlations between the two tables. The solution is found by maximizing the between-set dispersion, expressed by the covariance matrix between the two sets of variables, with respect to the within-set dispersion (Legendre and Legendre 1998, Section 11.4). The two sets of

variables must be quantitative and are assumed to be multinormally distributed. The limitation of the method is that the total number of variables in the two tables must be smaller than $(n-1)$.

In CCorA, one can also test the hypothesis of linear independence of the two multivariate data tables. Pillai and Hsu (1979) have shown that Pillai's trace is the most robust statistic to departures from normality.

The availability of RDA and CCA has limited the application of CCorA in ecology, since most ecological problems are stated in terms of control-response hypotheses for which asymmetrical ordination should be preferred. CCorA is more appropriate for exploratory purposes in cases where the two groups of variables are likely to influence each other, which may often occur in real ecological systems. Examples are the study of two groups of competing taxa and long-term studies of soil–vegetation relationships during a colonization process.

6.8.2 Canonical Correlation Analysis using CCorA

In the variation partitioning example of Sect. 6.3.2.7, we used two subsets of environmental variables, chemistry and physiography, to explain the structure of the fish data. Putting aside the variation partitioning of that example, we could study the structure of common variation of the two complete subsets of explanatory variables. How does chemistry relate to physiography?

Since the data should be as close as possible to the condition of multinormality, we transform some variables in the following example to make them more symmetrically distributed (we used the Shapiro–Wilk method to test for normality; results not shown here). The variables must be standardized since they have different physical dimensions. The function used for the analysis is **`CCorA()`** of package **`vegan`**.

```
# CANONICAL CORRELATION ANALYSIS (CCorA)
# ************************************

# Preparation of data (transformations to make variable
# distributions approximately symmetrical)
envchem2 <- envchem
envchem2$pho <- log(envchem$pho)
envchem2$nit <- sqrt(envchem$nit)
envchem2$amm <- log1p(envchem$amm)
envchem2$dbo <- log(envchem$dbo)

envtopo2 <- envtopo
envtopo2$alt <- log(envtopo$alt)
```

```
envtopo2$pen <- log(envtopo$pen)
envtopo2$deb <- sqrt(envtopo$deb)

# CCorA (on standardized variables)
chem.topo.ccora <- CCorA(envchem2, envtopo2, stand.Y=TRUE,
stand.X=TRUE, nperm=999)
chem.topo.ccora
biplot(chem.topo.ccora, plot.type="biplot")
```

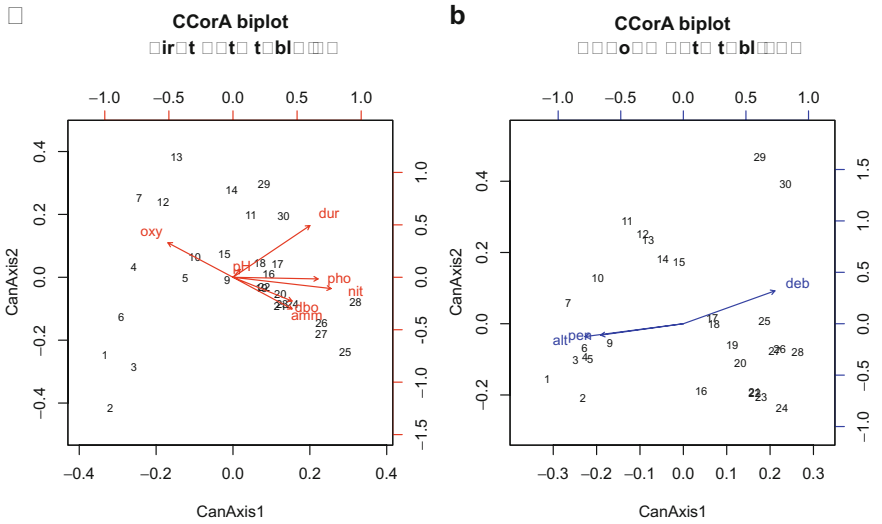

Fig. 6.10 Biplots of a canonical correlation analysis (CCorA) of the chemical (*left*) and physiographic (*right*) variables of the Doubs data

The result shows that there is a significant relationship between the two matrices (permutational probability = 0.001). Pillai's trace statistic is the sum of the squared canonical correlations. The canonical correlations are high on the first two axes. The RDA R^2 and adjusted R^2 are not part of the CCorA computations strictly speaking; the two RDAs are computed separately for information. This information is useful to assess whether the canonical axes are likely to express a substantial amount of variation (which is the case here), since canonical correlations may be large even when the common variation is small with respect to the total variation of the two data sets.

In Fig. 6.10 the left-hand biplot shows the standardized chemical variables and the objects projected in their space. The right-hand biplot shows the standardized physiographical variables and objects in their space. Note that the two spaces are "aligned" with respect to one another, i.e. the canonical axes show the same trends expressed by the two sets of variables. The positions of the sites in the two biplots are related, although not similar. The structures displayed are the result of *linear combinations* of variables in each of the biplots so that the interpretation of individual variables is difficult. The pair of biplots expresses the fact that oxygenated waters are related to high altitude and steep slope (i.e. upstream conditions), whereas discharge (`deb`) is highly positively correlated with hardness (`dur`), phosphates (`pho`) and nitrates (`nit`); altitude (`alt`) and slope (`pen`) are highly negatively correlated with these same variables.

CCorA is also available in package **`stats`** (function **`cancor()`**) and in a package (unfortunately) called **`CCA`**, a wrapper computing canonical correlations using **`cancor`** (in a function called **`cc()`**) and providing graphical outputs (function **`plt.cc()`**) and extensions to situations where the number of variables exceeds that of sites (function **`rcc()`**).

6.9 Co-inertia Analysis

6.9.1 Introduction

Dolédec and Chessel (1994) proposed an alternative to CCorA called CoIA. Dray et al. (2003) showed that this approach is a very general and flexible way to couple two or more data tables. CoIA is a symmetrical approach allowing the use of various methods to model the structure in each data matrix.

The analysis for two data tables is computed as follows:

- Compute the covariance matrix crossing the variables of the two data tables. The sum of squared covariances is the total co-inertia. Compute the eigenvalues and eigenvectors of that matrix. The eigenvalues represent a partitioning of the total co-inertia.
- Project the points and variables of the two original data tables on the co-inertia axes. By means of graphs, compare the projections of the two data tables in the common co-inertia space.

One particularly attractive feature of CoIA is the possibility to choose the type of ordination to apply to each data table prior to the joint analysis. Dray et al. (2003) gave examples of choices, each of them yielding different results. Of course the type of ordination must be chosen according to the research question and the mathematical type of the data. The main options are the ordination techniques explained in Chap. 5 (CA, PCA), but other methods can also be applied. Furthermore, within the intrinsic limits of the ordination methods applied to each data set, CoIA imposes fewer constraints than CCorA regarding the mathematical

type and the number of variables in the two tables. Note, however, that the row weights must be equal in the two separate ordinations, a condition that renders the use of CoIA with correspondence analysis difficult. CA is a weighted regression approach, and weights depend on the data. To apply CoIA, a choice must therefore be made about the weights of one or the two separate analyses to constrain them to be equal.

6.9.2 Co-inertia Analysis Using **ade4**

A function called **coinertia()** is available in package **ade4** to compute CoIA. In the code below, we apply CoIA to the chemical and physiographic subsets of environmental variables of the Doubs data set. Since **ade4** requires separate appropriate analyses, a PCA of the standardized data (correlation matrix) is first performed on each of the two data tables.[4] The number of axes to be retained in each analysis (2 by default) can be specified. The proportion of variance accounted for by the eigenvalues is then computed to assess the number of axes to be retained in the CoIA. In this example, three axes of the chemistry PCA account for 89.5% variation, and two axes of the physiography PCA account for 98.9% variation. After having verified that the row weights are equal in the two PCAs, these two results are then submitted to CoIA which is asked to retain two canonical axes. A permutation test is run to assess the significance of the co-structure of the data tables.

```
# CO-INERTIA ANALYSIS
# ******************

# PCA on both matrices
dudi.chem <- dudi.pca(envchem2, scale=TRUE, scan=FALSE, nf=3)
dudi.topo <- dudi.pca(envtopo2, scale=TRUE, scan=FALSE, nf=2)
# Relative variation of eigenvalues
dudi.chem$eig/sum(dudi.chem$eig)
# Relative variation of eigenvalues
dudi.topo$eig/sum(dudi.topo$eig)
# Equal row weights in the 2 analyses?
all.equal(dudi.chem$lw,dudi.topo$lw)

# Co-inertia analysis
coia.chem.topo <- coinertia(dudi.chem,dudi.topo, scan=FALSE,
  nf=2)
coia.chem.topo
```

[4] Note that **ade4** has been developed around a very general mathematical framework involving entities that will not be described here, called duality diagrams (Escoufier 1987). Readers are invited to consult the original publication to learn more about this framework.

```
# Relative variation on first eigenvalue
coia.chem.topo$eig[1]/sum(coia.chem.topo$eig)

summary(coia.chem.topo)
randtest(coia.chem.topo, nrepet=999)    # Permutation test
plot(coia.chem.topo)
```

Figure 6.11 gives a visual summary of the results of the CoIA. The numerical output looks like this:

```
Eigenvalues decomposition:
         eig     covar       sdX       sdY      corr
1 6.78049894 2.6039391 1.9994990 1.6364535 0.7958037
2 0.05644986 0.2375918 0.8714496 0.5354616 0.5091676

Inertia & coinertia X:
    inertia      max     ratio
1  3.997996 4.346012 0.9199230
12 4.757421 5.572031 0.8538037

Inertia & coinertia Y:
    inertia      max     ratio
1  2.677980 2.681151 0.9988172
12 2.964699 2.967525 0.9990479

RV:
 0.5537773
```

The numerical results first present the eigenvalue decomposition of the matrix of co-inertia: eigenvalues (*eig*), covariance (*covar*) and standard deviation (*sdX* and *sdY*) of the two sets of site scores on the co-inertia axes and correlations between the two sets of site scores. This correlation is computed as *covar*/(*sdX***sdY*).

The second block of results compares the inertia of the (cumulated) projections of the X and Y data tables as projected in the CoIA (*inertia*) compared to the maximum inertia of the axes of the separate ordinations (*max*). It also gives the ratio between these values as a measure of concordance between the two projections.

The RV coefficient is the ratio of the total co-inertia to the square root of the product of the squared total inertias of the separate analyses (Robert and Escoufier 1976). RV is a multivariate generalization of the Pearson correlation coefficient.

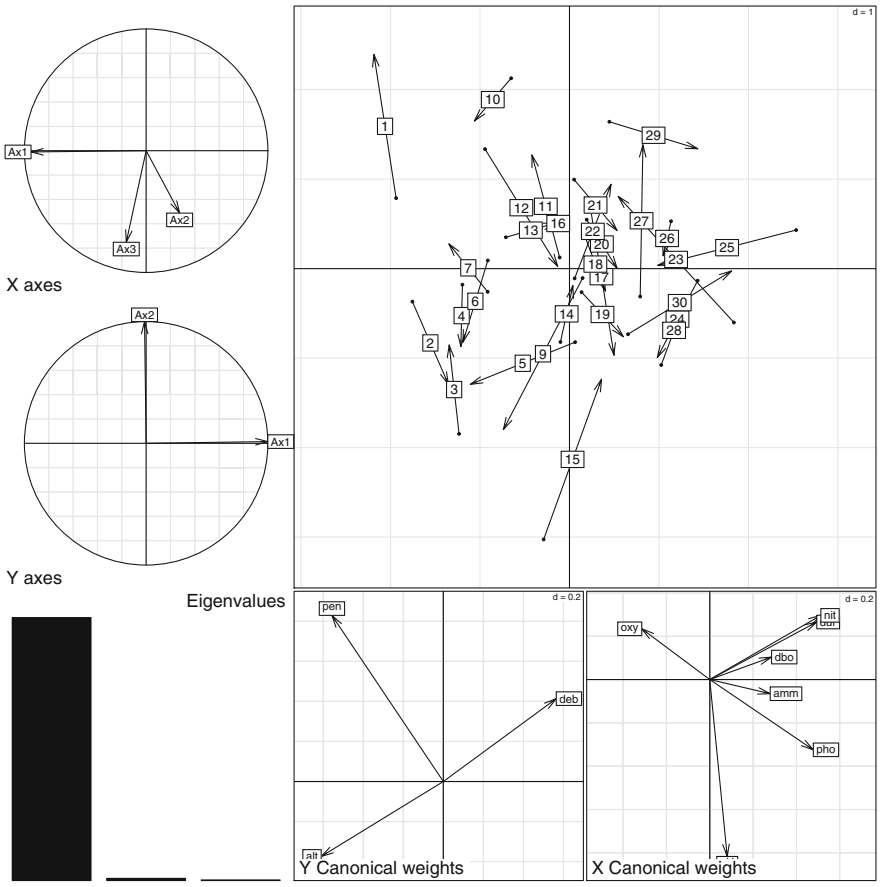

Fig. 6.11 Graphical results of a co-inertia analysis of the chemical and physiographical variables. Details: see text. X refers to the first and Y to the second data table

These results show that the first eigenvalue, representing 98.9% of the total variation, is overwhelmingly larger than the second one. Most of the common structure of the two data matrices is therefore to be sought on the first axis. The circular plots in Fig. 6.11 show that axes 1 of the two PCAs are almost perfectly aligned on the first CoIA analysis. The upper right-hand plot (normed site scores) shows the position of the sites on the co-inertia axes using the chemistry (origins of the arrows) and physiography (arrowheads) co-inertia weights. The shorter the arrows,

the better the concordance between the two projections. The lower right-hand pair of plots shows the contribution of the two groups of variables to the canonical space. Vectors pointing to the same direction are correlated and longer vectors contribute more to the structure. Oxygen (`oxy`) correlates positively with slope (`pen`), phosphates (`pho`) negatively with slope (`pen`); nitrates (`nit`), hardness (`dur`, label masked by nitrates) and biological oxygen demand (`dob`) are all negatively correlated with altitude (`alt`) since these variables have higher values downstream, and positively with discharge (deb, which increases downstream).

An extension of CoIA called RLQ analysis (Doledec et al. 1996; Dray et al. 2002) relates species traits to environmental variables by means of three tables: site-by-species (table L), site-by-environment (table R) and species-by-traits (table Q). It is available through the function **`rlq()`** of package **`ade4`**.

6.10 Multiple Factor Analysis

6.10.1 Introduction

Yet another approach to the symmetrical analysis of a data set described by k (usually $k>2$) subsets or groups of variables is multiple factor analysis (MFA; Escofier and Pagès 1994). This analysis is correlative; it excludes any hypothesis of causal influence of a data set on another. The variables must belong to the same mathematical type (quantitative or qualitative) within each subset. If all variables are quantitative, then MFA is basically a PCA applied to the whole set of variables in which each subset is weighted. MFA computation consists in the following steps:

- A PCA is computed for each (centred and optionally standardized) subset of variables. Each centred table is then weighted to give them equal weights in the global analysis, accounting for different variances among the groups. This is done by dividing all its variables by the first eigenvalue obtained from its PCA
- The k weighted data sets are concatenated. The resulting table is submitted to a global PCA
- The different subsets of variables are then projected on the global result; common structures and differences are assessed for objects and variables

The similarity between the geometrical representations derived from each group of variables is measured by the RV coefficient. RV coefficients, which vary between 0 and 1, can be tested by permutations (Josse et al. 2008).

MFA has been mainly used in sensory evaluation and chemistry so far, but the potential for ecological applications is promising, as evidenced by a few recent

contributions (Beamud et al. 2010; Carlson et al. 2010; Lamentowicz et al. 2010). Indeed, this method is very useful to explore the complex relationships among several ecologically meaningful groups of descriptors, whatever their number and type.

MFA can be run by using function **mfa()** of the package **ade4**. Note that a data frame comprising all blocks of variables must first be assembled and set into class ktab by function **ktab.data.frame()**. Here we shall use function **MFA()** of the package **FactoMineR** (Lê et al. 2008), which is more straightforward and offers more options.

6.10.2 Multiple Factor Analysis Using FactoMineR

In the code below, we apply MFA to three subsets of the Doubs data: the species (Hellinger-transformed abundances), the physiographical variables (upstream–downstream gradient), and the chemical variables (water quality). Note that this example is not at all equivalent to a constrained ordination, where the species data are explained by environmental variables and where the focus is put on an underlying, one-directional causal model. MFA proposes a symmetrical, exploratory point of view, where correlative structures are exposed without any reference to a directionality of possible causal relationships. No formal directional hypothesis is tested, either. This approach is therefore not adapted to the modelling of asymmetrical relationships, a task devoted to RDA or CCA. However, MFA could be used in early stages of a research project as a neutral data exploration technique to help generate causal hypotheses, which could be tested afterwards on an independent data set.

The function **MFA()** includes an important argument type, which allows specifying the mathematical type of each subset: "c" for continuous variables (to run a PCA on a covariance matrix), "s" for continuous variables requiring standardization (to run a PCA on a correlation matrix), or "n" for nominal variables (to run a MCA or multiple correspondence analysis, see Sect. 5.4.5). In our case, we have to state that the species subset belongs to type "c", whereas the two environmental subsets (chemistry and physiography) belong to type "s".

```
# MULTIPLE FACTOR ANALYSIS
# ************************

# MFA on 3 groups of variables

# Concatenate the 3 tables (Hellinger-transformed species,
# physiographical variables, chemical variables)
tab3 <- data.frame(spe.hel, envtopo, envchem)
dim(tab3)
(grn <- c(ncol(spe), ncol(envtopo), ncol(envchem)))
```

```
# Compute the MFA with multiple plots
t3.mfa <- MFA(tab3, group=grn, type=c("c","s","s"), ncp=2,
 name.group=c("Fish community","Physiography","Water quality"))
t3.mfa

plot(t3.mfa, choix="ind", habillage="none")
plot(t3.mfa, choix="ind", habillage="none", partial="all")
plot(t3.mfa, choix="var", habillage="group")
plot(t3.mfa, choix="axes")

# RV coefficients with tests (p-values above the diagonal of
# the matrix)
(rvp <- t3.mfa$group$RV)
rvp[1,2] <- coeffRV(spe.hel, scale(envtopo))$p.value
rvp[1,3] <- coeffRV(spe.hel, scale(envchem))$p.value
rvp[2,3] <- coeffRV(scale(envtopo), scale(envchem))$p.value
round(rvp[-4,-4], 6)

# Eigenvalues and % of variation
t3.mfa$eig
ev <- t3.mfa$eig[,1]
names(ev) <- 1:nrow(t3.mfa$eig)
evplot(ev)

# Select the most characteristic variables
aa <- dimdesc(t3.mfa, axes=1:2, proba=0.0001)

# Plot only the significant variables (correlations)
varsig <-
t3.mfa$quanti.var$cor[unique(c(rownames(aa$Dim.1$quanti),
  rownames(aa$Dim.2$quanti))),]
plot(varsig[,1:2], asp=1, type="n", xlim=c(-1,1), ylim=c(-1,1))
abline(h=0, lty=3)
abline(v=0, lty=3)
symbols(0, 0, circles=1, inches=FALSE, add=TRUE)
arrows(0, 0, varsig[,1], varsig[,2], length=0.08, angle=20)
for (v in 1:nrow(varsig)) {
  if (abs(varsig[v,1]) > abs(varsig[v,2])) {
    if (varsig[v,1] >= 0) pos <- 4
    else pos <- 2
    }
  else {
    if (varsig[v,2] >= 0) pos <- 3
    else pos <- 1
    }
```

```
  text(varsig[v,1], varsig[v,2], labels=rownames(varsig)[v],
pos=pos)
}
```

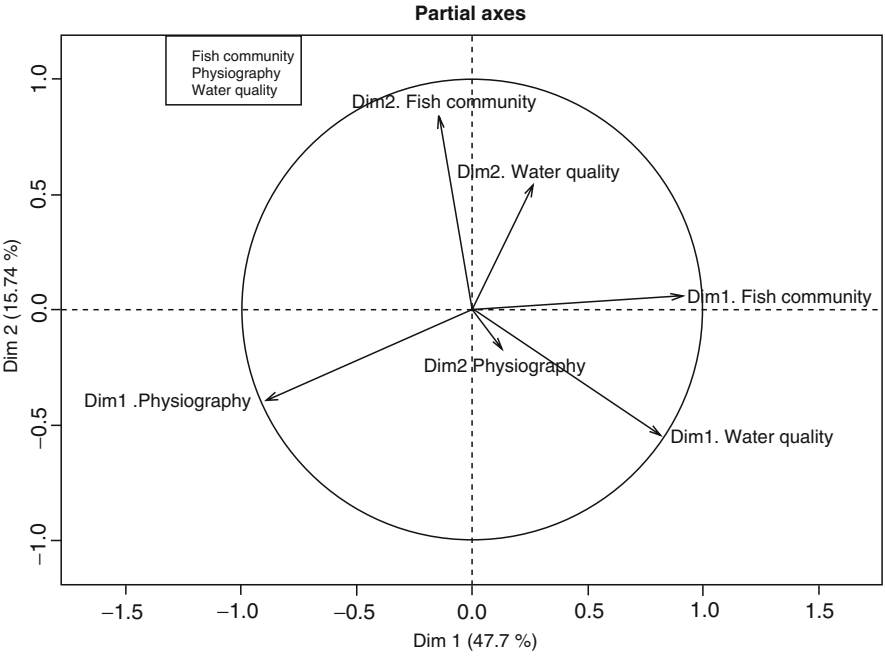

Fig. 6.12 Projection of the PCA axes of each subset on the MFA plane 1–2. The circle of radius 1 represents the maximum length of a partial standardized axis

The MFA provides an interesting picture of two main gradients and of the relationships among the three groups of variables. The first two axes represent more than 63% of the total variance. The plot "Partial axes" (Fig. 6.12) represents the projection of the principal components of each separate PCA on the global PCA. The plot "Individual factor map" (Fig. 6.13) shows the position of the sites according to four viewpoints: the labelled black points represent the MFA site scores (centroids of the site scores of each separate PCA); some of them are connected by coloured lines to the points representing their scores in the three separate PCAs.

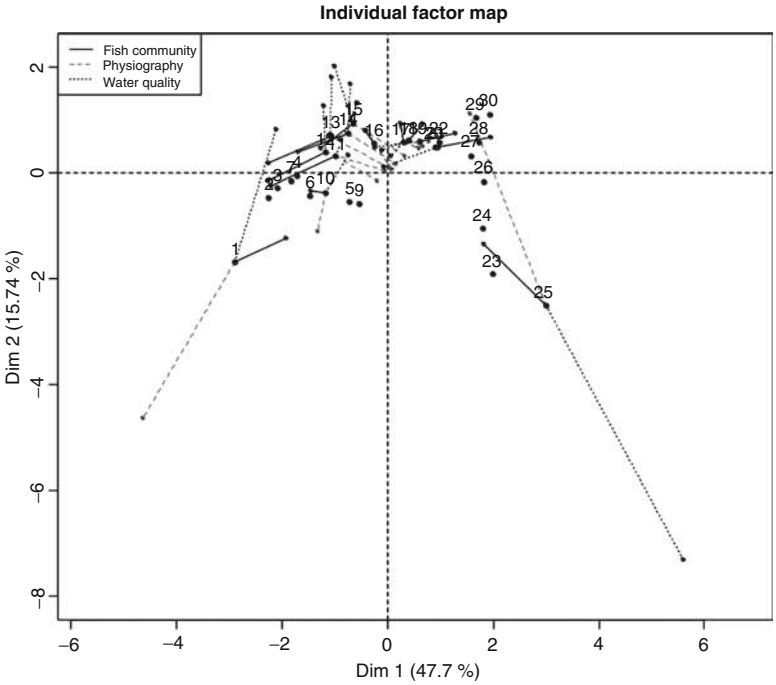

Fig. 6.13 MFA site scores (centroids; *black dots*) and selected PCA site scores with their links to the corresponding MFA centroids. For legibility not all PCA scores and links are drawn

The plot "Correlation circle" (Fig. 6.14) represents the normalized vectors of all quantitative variables.

In the table below, the RV coefficients appear in the lower-left triangle, below the diagonal, while the upper-right triangle contains permutational *p* values. These results tell us that fish communities are mostly linked to the physiographical conditions (RV = 0.58), which are themselves partly linked to water chemistry (RV = 0.36).

	Fish community	Physiography	Water quality
Fish community	1.000000	0.000002	0.000002
Physiography	0.580280	1.000000	0.002811
Water quality	0.505324	0.361874	1.000000

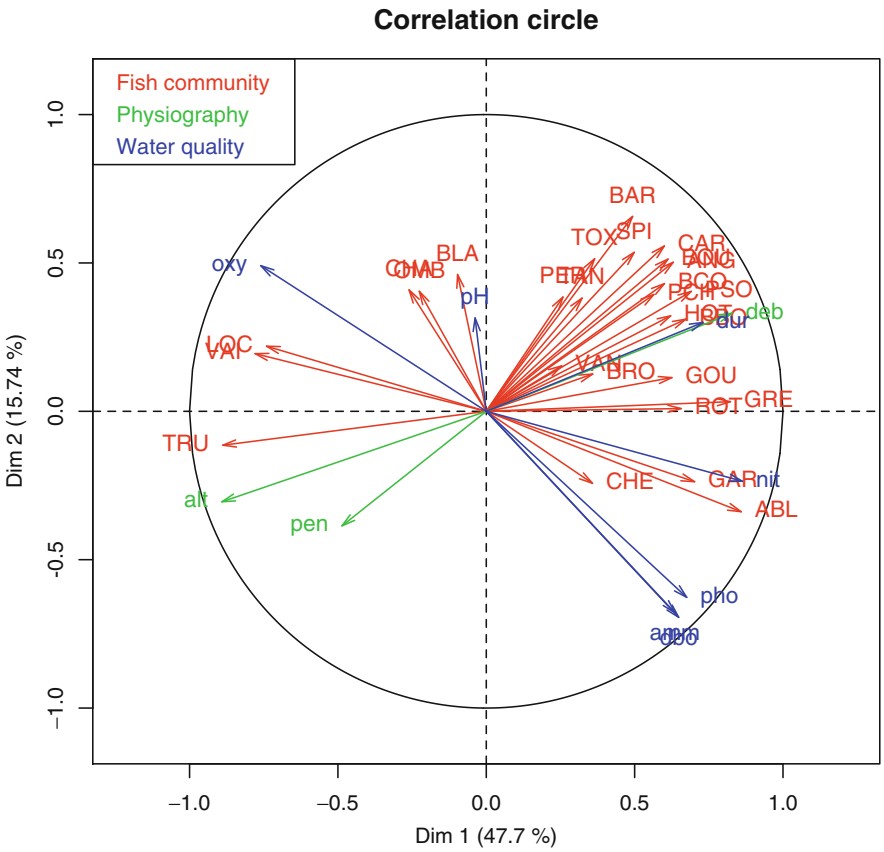

Fig. 6.14 Correlations between the variables of each subset and the MFA site scores on axes 1 and 2

If we examine Figs. 6.13 and 6.14 together, we can easily recognize the main upstream–downstream gradient along the first axis and the gradient of water quality along a combination of the first and second axes (from upper left to lower right). For example, the scores of sites 1, 2 and 3 (Fig. 6.13, left-hand part of the graph) correspond (Fig. 6.14) to a high altitude and a strong slope, as well as a high oxygen concentration. Here, close to the source, the ecological conditions are dominated by

physiography. The (relatively poor) fish community is characterized by `TRU`, `VAI` and `LOC`. On the opposite side, sites 23, 24 and 25 show the highest concentrations in phosphates, ammonium and nitrates, and a high biological oxygen demand. These three sites are heavily polluted and their community is characterized by another set of three species: `ABL`, `GAR` and `CHE`.

6.11 Conclusion

Ordination is a natural and informative way to look at ecological data, and ordination involving more than one data matrix is certainly the most powerful family of methods to reveal, interpret, test and model ecological relationships in the multivariate context. We presented what we think are the most important and useful methods, illustrating them using examples that highlighted their potential (Table 6.1). This is by no means the end of the story. Researchers continuously propose either new ways of exploiting the existing methods, or new methods altogether, whose merits and limits must be put under close scrutiny to allow them to be properly integrated into the already powerful statistical toolbox of the ecological community. Readers of this book are encouraged to join this movement.

Table 6.1 Names and some characteristics of the methods described in this chapter

Name, acronym	Use (examples)	**R** functions packages	Data; implementation; limitations
A. Asymmetric analyses			
Redundancy analysis, RDA	Predict **Y** with **X**: Variation partitioning	`rda` `{vegan}` `varpart` `{vegan}`	All types; species data with prior transformation; $m<(n-1)$. Linear model.
Canonical correspondence analysis, CCA	Predict **Y** with **X**	`cca` `{vegan}`	**Y**: species abundances; **X**: all types; $m < (n-1)$; unimodal response to latent variables
Linear discriminant analysis, LDA	Explain classification with quantitative variables	`lda` `{MASS}`	**Y**: classification; **X**: quantitative variables. Linear model.
Principal response curves, PRC	Model community response through time in controlled experiments	`prc` `{vegan}`	**Y**: community data; factor "treatment"; factor "time"
Space–time interaction analysis, STI	Test interaction in space–time data sets without replicated sites	`{STI}`	**Y**: community data; **T**: time vector; **S**: spatial coordinates
Co-correspondence analysis (asymmetric form), COCA	Predict one community on the basis of another	`coca` `{cocorresp}`	**Y**: data for community 1; **X**: data for community 2; both at the same sites. Unimodal response to latent variables

(continued)

Table 6.1 (continued)

Name, acronym	Use (examples)	**R** functions packages	Data; implementation; limitations
B. Symmetric analyses			
Co-correspondence analysis (symmetric form), COCA	Optimized comparison of two communities (descriptive approach)	`coca {cocorresp}`	$\mathbf{Y}_1$: data for community 1; $\mathbf{Y}_2$: data for community 2; both at the same sites. Unimodal response to latent variables
Canonical correlation analysis, CCorA	Common structures of two data matrices	`CCorA {vegan}`	Two matrices of quantitative data. Linear model
Co-inertia analysis, CoIA	Common structures of two or more data matrices	`coinertia {ade4}`	Very general and flexible; many types of data and ordination methods
RLQ analysis, RLQ	Species traits related to environmental variables	`rlq {ade4}`	Three tables: species-by-sites, species-by-environment; species-by-traits
Multiple factor analysis, MFA	Common structures of two or more data matrices	`mfa {ade4}` `MFA {FactoMineR}`	Simultaneous ordination of two or more weighted tables. Mathematical type must be homogeneous within each table

Chapter 7
Spatial Analysis of Ecological Data

7.1 Objectives

Spatial analysis of ecological data is a huge field that could fill several books by itself. To learn about general approaches in spatial analysis with **R**, readers may consult the recent book by Bivand et al. (2008). The present chapter has a more restricted scope. After a short general introduction, it deals with several methods that were specifically developed for the analysis of scale-dependent structures of ecological data; these methods can, of course, be applied to other domains. These methods are based on sets of variables describing spatial structures in various ways, derived from the coordinates of the sites or from the neighbourhood relationships among sites. These variables are used to model the spatial structures of ecological data by means of multiple regression or canonical ordination, and to identify significant spatial structures at all spatial scales that can be perceived by the sampling design. As you will see, the whole analytical process uses many of the techniques covered in the previous chapters.

Practically, you will:

- Learn how to compute spatial correlation measures and draw spatial correlograms
- Learn how to construct spatial descriptors derived from site coordinates and from links between sites
- Identify, test and interpret scale-dependent spatial structures
- Combine spatial analysis and variation partitioning
- Assess spatial structures in canonical ordinations by computing variograms of explained and residual ordination scores

D. Borcard et al., *Numerical Ecology with R*, Use R,
DOI 10.1007/978-1-4419-7976-6_7, © Springer Science+Business Media, LLC 2011

7.2 Spatial Structures and Spatial Analysis: A Short Overview

7.2.1 Introduction

As mentioned in Chap. 6, spatial structures play a very important role in the analysis of ecological data. Living communities are spatially structured at many scales, and these structures are the result of several classes of processes. On the other hand, beta diversity is the spatial variation in community composition; so, a study of the factors that can explain the spatial variation of community composition is in every respect an analysis of beta diversity. The environmental control model advocates that external forces (climatic, physical, chemical) control living communities. If these factors are spatially structured, their patterns will reflect on the living communities (examples: patches of desert where the soil is humid enough to support vegetation; gradient of successive communities through an intertidal zone). The biotic control model predicts that intra- and interspecific interactions within communities (examples: social groups of animals; top-down or bottom-up processes), as well as neutral processes such as ecological drift and limited dispersal, may result in spatial patterns which are the cause of spatial autocorrelation in the strict sense. Historical events (e.g. past disturbances like fire or human settlements) may have structured the environment in a way that still influences present-day communities.

In all, ecological data are a combination of many structures, spatial or not:

- The overall mean of each response variable; if the whole sampling area is under the influence of an all-encompassing process that changes the mean in a gradient across the area, then a *trend* is present. The trend may be due to a process operating at a scale larger than the sampling area.
- Spatial structures at regional scales: ecological processes of various kinds (biotic or abiotic) influence the data at scales finer than the overall sampling area, producing identifiable spatial patterns.
- Local deterministic structures with no recognizable spatial component because the sampling design is not fine enough to identify such fine-scale patches.
- Random noise (error): this is the residual (stochastic) component of the variation. It can be attributed to local effects operating independently at each sampling site.

One of the aims of spatial analysis is to discriminate between these sources of variation and model the relevant ones separately.

7.2.2 Induced Spatial Dependence and Spatial Autocorrelation

An important distinction must be made here. As we wrote above, a spatial structure in a response matrix **Y** can result from two main origins: either from the forcing of external (environmental) factors that are themselves spatially structured, or as the result of processes internal to the community itself. In the first case, one speaks of *induced spatial dependence*, in the second case of *spatial autocorrelation*.

For value y_j of a response variable **y** observed at site *j*, the model for *induced spatial dependence* is the following:

$$y_j = \mu_y + f(\mathbf{X}_j) + \varepsilon_j \tag{7.1}$$

where μ_y is the overall mean of variable **y**, **X** is a set of explanatory variables, and ε_j is an error term that varies randomly from location to location. The additional term [$f(\mathbf{X}_j)$] states that y_j is influenced by external processes represented in the model by explanatory variables. The spatial structure of these variables is reflected in *y*. When they form a gradient, they represent what Legendre (1993) called "true gradients", that is, gradient-like deterministic structures generated by external forces, whose error terms are not autocorrelated.

The model for *spatial autocorrelation* is:

$$y_j = \mu_y + \sum f(y_i - \mu_y) + \varepsilon_j \tag{7.2}$$

This equation states that y_j is influenced by the values of **y** at the surrounding sites *i*. This influence is modelled by a weighted sum of the (centred) values y_i at these sites. The biological context dictates the radius of the zone influencing a given point, as well as the weight to be given to the neighbouring points. These weights are generally dependent on the distance. The spatial interpolation method called *kriging* (Isaaks and Srivastava 1989; Bivand et al. 2008) is based on this model. Kriging is a family of interpolation methods that is not discussed further in this book. Kriging functions are available in package **geoR**.

Spatial autocorrelation may mimic gradients if the underlying process has a range of influence larger than the sampling area. Legendre (1993) called the resulting structures "false gradients". There is no statistical way to distinguish false from true gradients. One must rely upon biological hypotheses: in some cases, one has a strong hypothesis about the processes generating spatial structures, and therefore whether these processes may have produced autocorrelation in the data. In other cases, an opinion can be formed by comparing the processes detected at the scale

of the study area with those that are likely to occur at the scale of the (larger) target population (Legendre and Legendre 1998).

Spatial correlation measures the fact that near points in space have either more similar (positive correlation) or more dissimilar values (negative correlation) than randomly selected pairs. This phenomenon, which is generated either by true autocorrelation (7.2) or by spatial structures resulting from spatial dependence (7.1), has noxious effects on statistical tests. In spatially correlated data, values at any given site can be predicted, at least partially, from the values at other sites, if the researcher knows the biological process and the locations of the sites. This means that the values are not stochastically independent of one another. The assumption of independence of errors is violated in such cases. In other words, each new observation does not bring with it a full degree of freedom. While the fraction is difficult to determine, the fact is that the number of degrees of freedom used for a parametric test is often overestimated, thereby biasing the test on the "liberal" side: the null hypothesis is rejected too often. Numerical simulations have shown, however, that this statistical problem only occurs when both the response (e.g. species) and the explanatory variables (e.g. environmental) are spatially correlated (Legendre et al. 2002).

7.2.3 Spatial Scale

The term *scale* is used in many senses across different disciplines. It encompasses several properties of sampling designs and spatial analysis.

A sampling design has three characteristics pertaining to spatial scale (Legendre and Legendre 1998, Section 13.0):

- *Grain size*: size of the sampling units (diameter, surface or volume depending on the study).
- *Sampling interval*, sometimes called *lag*: average distance between neighbouring sampling units.
- *Extent* (sometimes called *range*): total length of the transect, surface area or volume (e.g. air, water) included in the study.

These three properties of a sampling design have an influence on the type and size of the spatial structures that can be identified and measured. (1) Sampling units *integrate* the structures occurring in them: one cannot identify structures of sizes equal to or smaller than the grain of the study. (2) The sampling interval determines the size of the finest spatial structures that can be identified (by *differentiation* among sampling units). (3) The extent of the study area sets an upper limit to the size of the measurable patterns. It is therefore essential to match each of these three

elements to the hypotheses to be tested and to the characteristics of the system under study (Dungan et al. 2002).

The ecological context of the study dictates the optimal grain size, sampling interval and extent. The optimal grain size (size of the sampling units) should match the size of unit entities of the study (e.g. objects like individual plants or animals, patches of vegetation, lakes, or areas affected by fine-scale processes). The average distance between unit objects or unit processes should be matched by the sampling interval. The extent should encompass the range of the broadest processes targeted by the study. These recommendations are detailed in Dungan et al. (2002).

Note that the expressions "large scale" and "small scale" are somewhat ambiguous because their meanings in ecology and cartography are opposite. In ecology "small scale" refers to the fine structures and "large scale" to the broadest structures, contrary to cartography where a large-scale map (e.g. 1:25000) is more detailed than a small-scale map (e.g. 1:1000000). Therefore, we advocate the use of "broad scale" (phenomena with large grains, large extents) and "fine scale" in ecology (Wiens 1989). Although these terms are not strict antonyms, we feel that they are less ambiguous than "large" and "small scale".

Finally, ecological processes occur at a variety of scales, resulting in complex, multiscale patterns. Therefore, identifying the scale(s) of the patterns and relating them to the appropriate processes are goals of paramount importance in modern ecology. To reach them, the researcher must rely on appropriate sampling designs and powerful analytical methods. The approaches presented in this chapter have been devised for the latter purpose.

7.2.4 Spatial Heterogeneity

A process or a pattern that varies across an area is said to be *spatially heterogeneous*. Many methods of spatial analysis are devoted to the measurement of the magnitude and extent of this heterogeneity and testing for the presence of spatial correlation (in other words, spatial structures of any kind). The latter may be done either to support the hypothesis that no spatial correlation (in the broad sense) is present in the data (if the researcher has classical parametric tests in mind) or, on the contrary, to show that correlation is present and use that information in conceptual or statistical models (Legendre and Legendre 1998).

Spatial heterogeneity in relation to inter-site distance is most often studied by means of *structure functions*. Examples of these are correlograms, variograms and periodograms. While it is not the purpose of this book to discuss these various functions, it is useful to devote a section to correlograms, since the main underlying measures of spatial correlation are used later in this chapter.

7.2.5 Spatial Correlation or Autocorrelation Functions and Spatial Correlograms

The two main statistics used to measure spatial correlation of univariate quantitative variables are Moran's *I* (Moran 1950) and Geary's *c* (Geary 1954). The first is constructed in much the same way as the Pearson correlation coefficient:

$$I(d) = \frac{\frac{1}{W}\sum_{h=1}^{n}\sum_{i=1}^{n} w_{hi}(y_h - \overline{y})(y_i - \overline{y})}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2} \quad \text{for} \quad h \neq i \tag{7.3}$$

The expected value of Moran's *I* for no spatial correlation is

$$E(I) = \frac{-1}{n-1} \tag{7.4}$$

Values below $E(I)$ indicate negative spatial correlation, and values above $E(I)$ indicate positive correlation. $E(I)$ is close to 0 when n (the total number of observations) is large.

Geary's *c* is more akin to a distance measure:

$$c(d) = \frac{\frac{1}{2W}\sum_{h=1}^{n}\sum_{i=1}^{n} w_{hi}(y_h - y_i)^2}{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2} \quad \text{for} \quad h \neq i \tag{7.5}$$

The expected value of Geary's *c* for no spatial correlation is $E(c) = 1$. Values below 1 indicate *positive* spatial correlation, and values above 1 indicate *negative* correlation.

y_h and y_i are the values of variable y at pairs of sites h and i. To compute spatial correlation coefficients, one first constructs a matrix of geographical distances among sites. These distances are then converted to classes d. Both formulas show the computation of the index value for a class of inter-site distance d. The weights w_{hi} have value $w_{hi} = 1$ for pairs of sites belonging to distance class d, and $w_{hi} = 0$ otherwise. W is the number of pairs of points used to compute the coefficient for the distance class considered, i.e., the sum of the w_{hi} weights for that class.

A *correlogram* is a plot of the spatial correlation values against the distance classes. Combined with statistical tests, a correlogram allows a quick assessment of the type and range of the spatial correlation structure of a variable. A typical case is spatial correlation that is positive at short distances, decreases to negative values, and levels out to a point where it becomes non-significant. The corresponding distance class sets the distance beyond which a pair of values can be considered as spatially independent. It is important to note that spatial correlograms display any kind of spatial correlation, generated by (7.1) (induced spatial dependence) or (7.2) (spatial autocorrelation); so the name "spatial autocorrelogram" which is often given to these plots is somewhat misleading.

Univariate spatial correlograms can be computed using the function **`sp.correlogram()`** of package **`spdep`**. We can apply this function to the variable "Substrate density" of the oribatid mite data set. We first define neighbourhoods of size ≤ 0.7 m around the points using the function **`dnearneigh()`**. These links can be visualized using our function **`plot.links()`**. Following that, the function **`sp.correlogram()`** finds successive lag orders of contiguous neighbours and computes Moran's *I* for each of these lag orders. A lag order is the number of links, or steps in the linkage graph, between two points. It can be construed as a generalized form of distance between points. For instance, if sites A and C are connected through site B, two links (A–B and B–C) are needed to connect A and C. They are connected at lag order 2.

Note: Cartesian coordinates can be obtained from latitude–longitude (sometimes abbreviated to Lat/Lon or LatLon) data using the function **`geoXY()`** of package **`SoDA`**.

```
# Load the required packages
library(ape)
library(spdep)
library(vegan)
library(ade4)

# Packages available at the following URL:
# https://r-forge.r-project.org/R/?group_id=195
library(packfor)
library(spacemakeR)
library(AEM)
library(PCNM)
source("plot.links.R")  # Function must be in working directory
source("sr.value.R")    # Function must be in working directory
```

```
# Import the data

mite <- read.table("mite.txt")
mite.env <- read.table("mite_env.txt")
mite.xy <- read.table("mite_xy.txt")

mite.h <- decostand (mite, "hellinger")
mite.xy.c <- scale(mite.xy, center=TRUE, scale=FALSE)

# Spatial correlogram (based on Moran's I)
# ****************************************
# Search for neighbours of all points within a radius of 0.7 m
# and multiples (i.e., 0 to 0.7m, 0.7 to 1.4m and so on). The
# points do not form a connected graph at 0.7 m.
plot.links(mite.xy, thresh=0.7)
nb1 <- dnearneigh(as.matrix(mite.xy), 0, 0.7)
summary(nb1)

# Correlogram of substrate density
subs.dens <- mite.env[,1]
subs.correlog <- sp.correlogram(nb1, subs.dens, order=14,
method="I", zero.policy=TRUE)
print(subs.correlog, p.adj.method="holm")
plot(subs.correlog)
```

Hint *We use the* **`print()`** *function to display the correlogram results because it allows for correction of the p values for multiple testing. In a correlogram, a test is performed for each lag (distance class), so that without correction, the overall risk of type I error is greatly increased. The Holm (1979) correction is applied here.*

This correlogram has a single significant distance class: there is positive spatial correlation at distance class 1 (i.e. 0.0–0.7 m). Negative spatial correlation at distance class 4 (i.e. 2.1–2.8 m) is hinted at, but the coefficient is not significant after Holm (1979) correction for multiple testing (see Sect. 7.2.6). Beyond this mark, no significant spatial correlation is identified, which means that for practical purposes measurements taken more than 0.7 m, or (conservatively) 2.8 m apart (the upper limit of class 4), can be considered as spatially independent with respect to substrate density.

Spatial correlation in the multivariate domain can be assessed and tested for by means of a *Mantel correlogram* (Sokal 1986; Oden and Sokal 1986). Basically, one computes a normalized Mantel statistic r_M (analogous to a Pearson's r coefficient) between a dissimilarity matrix among sites and a matrix where pairs of sites belonging to the same distance class receive value 0 and the other pairs, value 1. The process

is repeated for each distance class. Each r_M value can be tested by permutations. The expectation of the Mantel statistic for no spatial correlation is $r_M = 0$.

A Mantel correlogram can be computed, tested and plotted (Fig. 7.1) by using **vegan**'s function **mantel.correlog()**. The only data necessary are a response distance matrix and either the geographical coordinates of the sites or a matrix of geographical distances among sites. Here is an example of a Mantel correlogram for the oribatid mite data, which is first detrended (Sect. 7.3.2) to make the data second-order stationary (Sect. 7.2.6).

```
# Mantel correlogram of the oribatid mite data
# ********************************************

# The species data are first detrended; see Section 7.3
mite.h.det <- resid(lm(as.matrix(mite.h) ~ ., data=mite.xy))

mite.h.D1 <- dist(mite.h.det)
(mite.correlog <- mantel.correlog(mite.h.D1, XY=mite.xy,
nperm=99))
summary(mite.correlog)
plot(mite.correlog)
```

Hint *In this run, the number of classes has been computed automatically using Sturge's rule. Use argument* `n.class` *to provide a user-determined number of classes.*

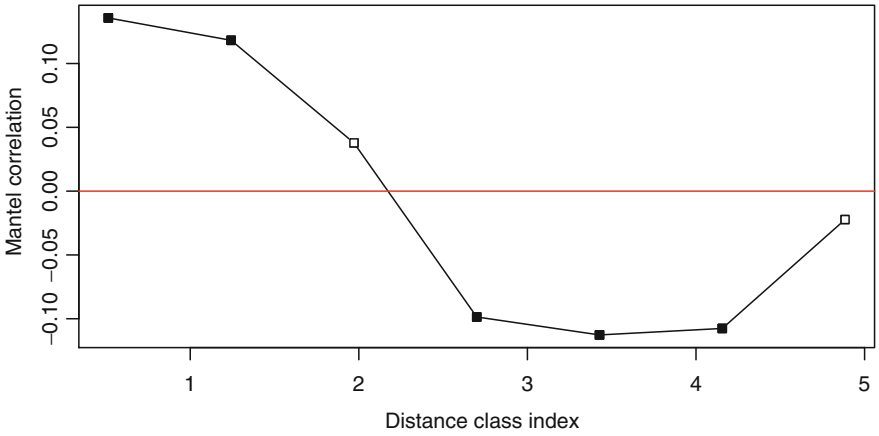

Fig. 7.1 Mantel correlogram of the Hellinger-transformed and detrended oribatid mite species data. *Black squares* indicate significant multivariate spatial correlation after Holm correction for multiple testing. The abscissa is labelled in metres since this is the unit of the data used to construct the distance classes

In this simple run, most default settings have been applied, including Holm's correction for multiple testing (see Sect. 7.2.6). The number of classes has been computed using Sturge's rule [number of classes = 1 + (3.3219 × $\log_{10} n$), where n is the number of elements, here the number of pairwise distances]. The resulting number of classes and the corresponding break points can be read in the result object:

```
# Number of classes
mite.correlog$n.class       # or: mite.correlog[2]
# Break points
mite.correlog$break.pts     # or: mite.correlog[3]
```

Hint *The default option* `cutoff=TRUE` *limits the correlogram to the distance classes including all points (the first seven distance classes in this example); the results for the last five distance classes (computed on fewer and fewer points) are not shown.*

The result shows significant positive spatial correlation in the first two distance classes (i.e. between 0.15 and 1.61 m; see the break points) and negative significant correlation in the fourth to sixth classes (between 2.34 and 4.52 m). Examining the environmental variables allows some speculation about the ecological reasons behind these structures. Close sites tend to show similar communities because the soil conditions are rather similar. On the other hand, any pair of sites whose members are about 2.7 m apart (class index of distance class 4) falls into contrasting soil conditions, which in turn explains why their mite communities are different.

7.2.6 Testing for the Presence of Spatial Correlation: Conditions

As shown above, spatial correlation coefficients can be tested for significance. However, conditions of application must be respected. The condition of normality can be relaxed if the test is carried out by permutations. To test the significance of coefficients of spatial correlation, however, the condition of *second-order stationarity* must be met. That condition states that the mean of the variable and its spatial covariance (numerator of (7.3)) are the same over the study area, and that its variance (denominator of (7.3)) is finite. This condition tells us, in other words, that the spatial variation of the data should be adequately described by the same single spatial correlation function in all portions of the study area. Spatial correlation coefficients cannot be tested for significance if an overall trend is present in the data ("true gradient"), or if the variable has been measured in a region where several distinct structures should be modelled by different spatial correlation

functions. Data displaying simple trends can often be *detrended* by means of a first- or higher-degree function of the site geographical coordinates (Sect. 7.3).

Another, relaxed form of stationarity is called the *intrinsic assumption*, a short form for "hypothesis of intrinsic stationarity of order 2" (Wackernagel 2003). This condition considers only the increments of the values of the variable; it states that the differences $(y_h - y_i)$ for any distance d (in the numerator of (7.5)) have zero mean and constant and finite variance over the study area, independently of the location (Legendre and Legendre 1998). This condition allows one to compute and examine correlograms but without tests of significance.

Legendre and Legendre (1998, p. 721) show how to interpret all-directional correlograms (i.e. correlograms built on distance classes defined the same way in all directions) as well as directional correlograms.

A word is needed here about *multiple testing*. In Sect. 7.2.5 several spatial correlation values were tested simultaneously for significance. In such cases, the probability of type I error increases with the number of tests. If k tests are carried out, the binomial law tells us that the overall probability of type I error (technically called the "experimentwise error rate") is equal to $1-(1-\alpha)^k$ where α is the nominal value for a single test. For instance, in the Mantel correlogram shown in Fig. 7.1, seven tests are carried out simultaneously. Without correction, the *overall* probability of obtaining *at least* one type I error is equal to $1-(1-0.05)^7 = 0.302$ instead of the nominal $\alpha = 0.05$. Several methods have been proposed to achieve a correct level of type I error in multiple tests (reviewed in Legendre and Legendre 1998; Wright 1992). The most conservative solution for k independent tests is to divide the significance level by the number of simultaneous tests: $\alpha' = \alpha/k$ and compare the p values to α'. Conversely, one can multiply the p values by k (i.e. $p' = kp$) and compare the resulting values to the unadjusted α. For non-independent tests, Holm's procedure (Holm 1979) is more powerful. The reason is that Holm's correction consists in applying Bonferroni's correction sequentially by progressively relaxing the correcting factor as follows. First, order the (uncorrected) p values in increasing order from top to bottom. Then, multiply the smallest p value by k, the second smallest by $k-1$, and so on. If an adjusted p value is smaller than the previous one, make it equal to it. Compare the resulting values to the unadjusted alpha.

Other corrections have been proposed in addition to the two presented above. Several are available in a function called **`p.adjust()`** in package **`stats`**. This function can be called whenever one has run several simultaneous tests of significance. The data submitted to that function must be a vector.

7.2.7 Modelling Spatial Structures

Beyond the methods described above, there are other, more modelling-oriented approaches to spatial analysis. Finding spatial structures in ecological data indicates that some process has been at work to generate them; the most important are environmental forcing (past or present) and biotic processes. Therefore, it is interesting to identify the spatial structures in the data and model them. Spatial structures can then either be related to explanatory variables representing hypothesized causes, or help generate new hypotheses as to which processes may have generated them.

Spatial structures can be present at many different scales. Identifying these scales and modelling the corresponding spatial structures separately is a long-sought goal for ecologists. A first, rather coarse approach in multivariate analysis is the adaptation of trend-surface analysis to canonical ordination. As suggested by ter Braak (1987) and demonstrated by Legendre (1990), response data may be explained by a polynomial function of the (centred) site coordinates. Borcard et al. (1992) have shown how to integrate this method into variation partitioning to identify, among other fractions, the pure spatial component of the ecological variation of species assemblages.

Multivariate trend-surface analysis only allows one to extract rather simple spatial structures because polynomial terms become rapidly cumbersome, and highly correlated if one uses raw polynomials. In practice, their use is restricted to third-degree polynomials. A breakthrough came with the development of principal coordinates of neighbour matrices (PCNM) and other forms of eigenvector-based spatial functions, which are described in Sect. 7.4, after a short example of trend-surface analysis.

7.3 Multivariate Trend-Surface Analysis

7.3.1 Introduction

Most ecological data have been sampled on geographic surfaces. Therefore, the crudest way to model the spatial structure of the response data is to regress them on the *X*–*Y* coordinates of the sampling sites. Of course, this will only model a *linear trend*; a plane will be fitted through the data in the same way as a straight line would be fitted to data collected along a transect by regressing them on their *X* coordinates.

A way of allowing curvilinear structures to be modelled is to add polynomial terms of the coordinates to the explanatory data. Second- and third-degree terms are often applied. It is better to centre (but not standardize, lest one distort the aspect-ratio

of the sampling design) the X and Y coordinates before computing the polynomial terms, to make at least the second-degree terms less correlated. The first-, second- and third-degree functions are:

$$\hat{z} = f(X,Y) = b_0 + b_1 X + b_2 Y \tag{7.6}$$

$$\hat{z} = b_0 + b_1 X + b_2 Y + b_3 X^2 + b_4 XY + b_5 Y^2 \tag{7.7}$$

$$\hat{z} = b_0 + b_1 X + b_2 Y + b_3 X^2 + b_4 XY + b_5 Y^2 + b_6 X^3 + b_7 X^2 Y + b_8 XY^2 + b_9 Y^3 \tag{7.8}$$

An alternative method is to compute orthogonal polynomial terms using the function **poly()** with the default option `raw=FALSE`, which produces orthogonal polynomials. For a set of X–Y coordinates, the monomials X, X^2, X^3 and Y, Y^2, Y^3 have a norm of 1 and are orthogonal to their respective lower order terms. X monomials are not orthogonal to Y monomials, however, except when the points form a regular orthogonal grid; terms containing both X and Y are not orthogonal to one another and their norms differ from 1. Orthogonal polynomials produce the exact same R^2 in regression and canonical analysis as raw polynomials. The orthogonality of orthogonal polynomials presents an advantage when selection of explanatory variables is used to find a parsimonious spatial model.

Trend-surface analysis can be applied to multivariate data by means of RDA or CCA. The result is a set of independent spatial models (one for each canonical axis). One can also use forward selection to reduce the model to its significant components only.

7.3.2 Trend-Surface Analysis in Practice

Our first step in spatial modelling is to produce some monomials of the X and Y coordinates on a grid just to become familiar with the shapes they produce through visualization. We then proceed to apply this technique to the oribatid mite data. As a courtesy to our readers, we have modified **ade4**'s **s.value()** function to draw round instead of square bubbles in some plots. The modified function is called **sr.value()**.

```
# Trend-surface analysis
# **********************

# Simple models on a square, regularly sampled surface

# Construct and plot a 10 x 10 grid
xygrid <- expand.grid(1:10, 1:10)
plot(xygrid)
```

```
xygrid.c <- scale(xygrid, scale=FALSE)  # Centring
X <- xygrid.c[,1]
Y <- xygrid.c[,2]

# Plot some first, second and third-degree functions of X and Y
par(mfrow=c(3,3))
s.value(xygrid,(X))
s.value(xygrid,(Y))
s.value(xygrid,(X+Y))
s.value(xygrid,(X^2+Y^2))
s.value(xygrid,(X^2-X*Y-Y^2))
s.value(xygrid,(X+Y+X^2+X*Y+Y^2))
s.value(xygrid,(X^3+Y^3))
s.value(xygrid,(X^3+X^2*Y+X*Y^2+Y^3))
s.value(xygrid,(X+Y+X^2+X*Y+Y^2+X^3+X^2*Y+X*Y^2+Y^3))

  # Try other combinations, for instance with minus signs or with
  # coefficients not equal to 1.

# Trend-surface analysis of the mite data
# ***************************************
# Computation of the standard (non-orthogonal) third-degree
polynomial
# function on the previously centred X-Y coordinates
mite.poly <- poly(as.matrix(mite.xy.c), degree=3, raw=TRUE)
colnames(mite.poly) <-
  c("X","X2","X3","Y","XY","X2Y","Y2","XY2","Y3")

  # Function poly produces the polynomial terms in the following
  # sequence: X, X^2, X^3, Y, XY, X^2Y, Y^2, XY^2, Y^3).
  # The original column names give the degree for the two
  # variables.
  # For instance, "1.2" means X^1*Y^2.
  # Here raw polynomials have been computed. For orthogonal
  # polynomials, which is the default, raw=FALSE.

# RDA with all 9 polynomial terms
mite.trend.rda <- rda(mite.h ~ ., data=as.data.frame(mite.poly))

# Computation of the adjusted R^2
(R2adj.poly <- RsquareAdj(mite.trend.rda)$adj.r.squared)

# RDA using a third-degree orthogonal polynomial of the
# geographic coordinates
mite.poly.ortho <- poly(as.matrix(mite.xy), degree=3)
```

```
colnames(mite.poly.ortho) <-
 c("X","X2","X3","Y","XY","X2Y","Y2","XY2","Y3")
mite.trend.rda.ortho <- rda(mite.h~.,
data=as.data.frame(mite.poly.ortho))
(R2adj.poly <- RsquareAdj(mite.trend.rda.ortho)$adj.r.squared)

# Forward selection using Blanchet et al. (2008a) double
# stopping criterion
mite.trend.fwd <- forward.sel(mite.h, mite.poly.ortho,
adjR2thresh=R2adj.poly)

# New RDA using the 6 terms retained
(mite.trend.rda2 <- rda(mite.h ~ .,
data=as.data.frame(mite.poly)[,mite.trend.fwd[,2]]))

# Overall test and test of the canonical axes
anova.cca(mite.trend.rda2, step=1000)
anova.cca(mite.trend.rda2, step=1000, by="axis")

# Plot of the three independent significant spatial structures
# (canonical axes). For square bubbles type "s.value" instead of
# "sr.value".
mite.trend.fit <- scores.cca(mite.trend.rda2, choices=c(1,2,3),
display="lc", scaling=1)
par(mfrow=c(1,3))
sr.value(mite.xy,mite.trend.fit[,1])
sr.value(mite.xy,mite.trend.fit[,2])
sr.value(mite.xy,mite.trend.fit[,3])

# ------------------------------------------------------------

 # If you want to construct a raw polynomial function directly
 # within the rda call, here is the syntax (2nd degree):
 # mite.trend.rda <- rda(mite.h ~ Xm + Ym + I(Xm^2) + I(Xm*Ym)
 # + I(Ym^2))
 # Notice how squared variables and product variables are
 # requested to be treated "as they are" by function I().
 # Otherwise R would consider them as ANOVA terms.
```

Hint *Note that the **fitted site scores in scaling 1** have been used in the plots. We want to display the "pure" spatial model, i.e. the linear combination of spatial variables, in a projection preserving the Euclidean distances among sites.*

This analysis shows that the oribatid mite community is significantly spatially structured, and that three significant independent models can be obtained. The first one (first canonical axis, 73.8% of the *explained* variance) displays a strong difference between the upper and the lower half of the area. The two other models (12.1 and 8.5% of the explained variance, respectively) display finer-scale structures (Fig. 7.2).

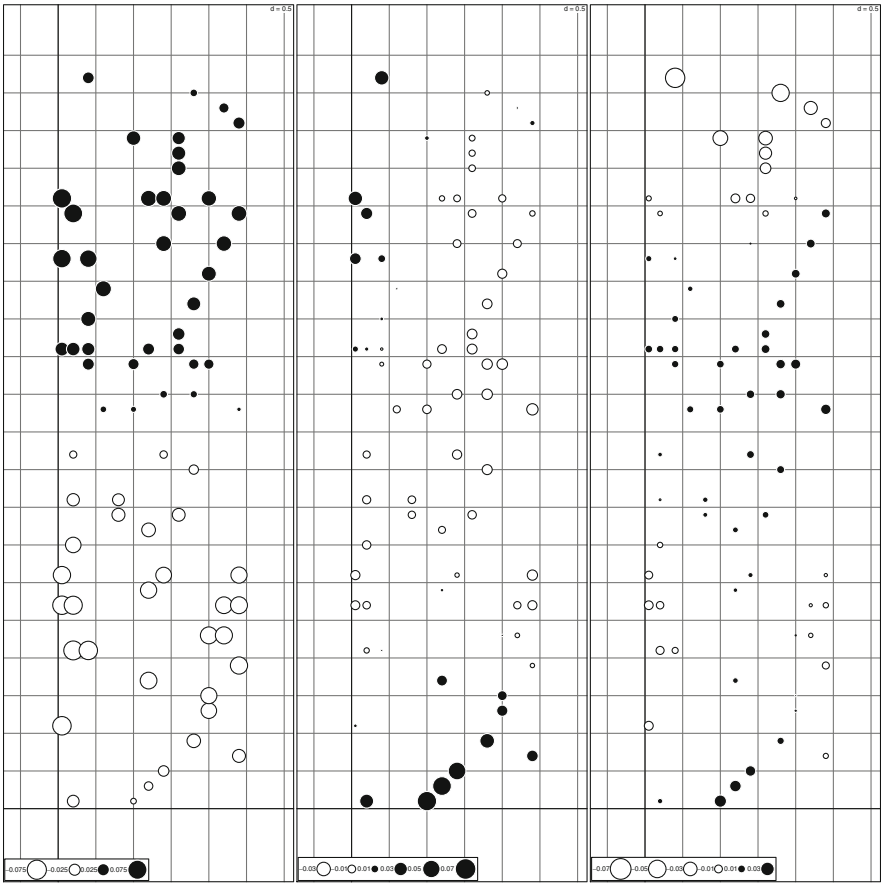

Fig. 7.2 Cubic trend-surface analysis of the Hellinger-transformed oribatid mite data. Three significant RDA axes have been retained, representing linearly independent spatial structures

These models could now be interpreted by regressing them on environmental variables. But we postpone that step until we can implement it in another spatial modelling framework.

Nowadays, the most useful application of trend-surface analysis is for *detrending*. We have seen in Sect. 7.2.6 that data have to be detrended before spatial correlograms can be tested. We will also see later that eigenvector-based spatial analyses are best applied to detrended data. Therefore, a handy procedure is to test for linear trends and detrend the data if the trend surface is significant. This means to regress all variables on the *X*–*Y* coordinates and retain the residuals. This can most easily be done using the function **`lm()`**.

```
# Detrending the mite data
# ************************
anova(rda(mite.h, mite.xy))  # Result: significant trend
# Computation of linearly detrended mite data
mite.h.det <- resid(lm(as.matrix(mite.h) ~ ., data=mite.xy))
```

This detrended data set is now ready for more complex spatial analyses and modelling.

7.4 Eigenvector-Based Spatial Variables and Spatial Modelling

7.4.1 Introduction

Trend-surface analysis is a rather coarse method of spatial modelling. The multi-scale nature of ecological processes and data calls for other approaches that can identify and model structures at all scales that can be perceived by a sampling design. Practically, this means methods that could model structures at scales ranging from the broadest, encompassing the whole sampled area, down to the finest, whose sizes are of the order of magnitude of the sampling interval. To achieve this in the context of canonical ordination, we must construct spatial variables representing structures of all relevant scales. This is what the PCNM method (principal coordinates of neighbour matrices; Borcard and Legendre 2002; Borcard et al. 2004) and its offsprings do. These methods will now be studied in detail.

As will be shown in Sect. 7.4.3, the PCNM method is actually a special case of a wider family of methods that are now called MEM (Moran's eigenvector maps; Dray et al. 2006). However, since many published papers cite this variant under its

original name, we use it here for explanatory purposes. Bear in mind, however, that the acronym PCNM is likely to be short-lived in the literature. New papers use the generic acronym MEM.

7.4.2 Classical Distance-Based MEM, Formerly Called Principal Coordinates of Neighbour Matrices

7.4.2.1 Introduction

The PCNM method works as follows:

- Construct a matrix of Euclidean (geographic) distances among sites.
- Truncate this matrix to retain only the distances among close neighbours. The threshold depends on the data. In most cases, it is chosen to be as short as possible, but all points must remain connected by links smaller than or equal to the truncation distance. Otherwise, different groups of eigenfunctions are created, that model the spatial variation within separate subgroups of points but not among these groups. How to choose the truncation threshold distance is discussed below. All pairs of points more distant than the threshold receive an arbitrary "large" distance value corresponding to four times the threshold.
- Compute a PCoA of the truncated distance matrix.
- In most studies, retain the eigenvectors that model positive spatial correlation (Moran's I larger than $E(I)$, (7.4)).
- Use these eigenvectors as spatial explanatory variables in multiple regression or RDA.

The PCNM method presents several advantages over trend-surface analysis. It produces *orthogonal* (linearly independent) spatial variables over a *much wider range* of spatial scales. It allows the modelling of *any type* of spatial structures, as Borcard and Legendre (2002) have demonstrated through extensive simulations.

The PCNM method can work for any sampling design, although the spatial variables are easier to interpret in the case of regular designs, as shown below. When the design is irregular, it may happen that a large truncation value must be chosen to allow all points to remain connected. A large truncation value means a loss of the finest spatial structures. Therefore, ideally, even an irregular sampling design should ensure that the minimum distance allowing all points to be connected is as short as possible. In cases where this distance is too large, Borcard and Legendre (2002) suggested (1) to add a limited number of supplementary points to the spatial data to cut down the threshold distance, (2) compute the PCNM variables, and (3) remove the supplementary points from the PCNM matrix. This ensures that

the finest scales are better modelled. The trade-off is that the resulting PCNM variables are no longer totally orthogonal to one another, but if the number of supplementary points is small with respect to the number of true points, the departure from orthogonality remains small.

The classical PCNM method produces eigenfunctions for all positive eigenvalues. However, some of these eigenfunctions display negative spatial correlation. In most studies, one is primarily interested in patterns produced by spatially contagious processes, which display positive spatial correlation. Therefore, it is generally preferable to retain only the eigenfunctions with Moran's $I > E(I)$, or to run separate analyses with the eigenfunctions with positive and negative spatial correlation. The relationship between the sign of the eigenvalues and the sign of the spatial correlation is not simple for PCNM, whereas the value of Moran's I is a linear function of the eigenvalue in the case of standard MEM eigenfunctions. So it is advised to compute Moran's I in all cases. Function **`PCNM()`** of the package **`PCNM`** presented in Sect. 7.4.2.3 can do that automatically (argument `moran`).

7.4.2.2 PCNM Variables on Regular Sampling Designs

When the spatial coordinates correspond to points that are equispaced along a transect or across a surface, the resulting PCNM variables represent a series of sinusoids of decreasing periods. For a transect with n regularly spaced points and sampling interval s, the wavelength λ_i of the eigenfunction with rank i is: $\lambda_i = 2(n+s)/(i+1)$ (Guénard et al. 2010, eq. 3).[1] Let us construct and illustrate a one-dimensional (Fig. 7.3) and a two-dimensional (Fig. 7.4) example, both equispaced.

```
# Constructing PCNM variables step by step
# ****************************************

# 1. One-dimensional sampling: transect with 100 equispaced
#    points. The distance between adjacent points is 1

tr100 <- seq(1:100)             # Generate transect points
tr100.d1 <- dist(tr100)         # Euclidean distance matrix
thresh <- 1                     # truncation distance set to 1
# Truncation to threshold 1
tr100.d1[tr100.d1 > thresh] <- 4*thresh
# PCoA of truncated matrix
tr100.PCoA <- cmdscale(tr100.d1, eig=TRUE, k=length(tr100)-1)
```

[1] A simple function to find the wavelength for an intersite distance $s = 1$ is: `wavelength <- function(i,n) {2*(n+1)/(i+1)}`.

```
# Count the positive eigenvalues
(nb.ev <- length(which(tr100.PCoA$eig > 0.0000001)))
# Matrix of PCNM variables
tr100.PCNM <- tr100.PCoA$points[,1:nb.ev]

# Plot some PCNM variables modelling positive spatial
# correlation (Fig. 7.3)
par(mfrow=c(4,2))
somePCNM <- c(1, 2, 4, 8, 15, 20, 30, 40)
for(i in 1:length(somePCNM)){
 plot(tr100.PCNM[,somePCNM[i]], type="l", ylab=c("PCNM",
  somePCNM[i]))
}

# 2. Two-dimensional sampling: equispaced grid
#    The truncation distance is set to 1. It could also be
#    chosen to be the diagonal distance within a small square
#    of 4 points, sqrt(2)

xygrid2 <- expand.grid(1:20, 1:20)
xygrid2.d1 <- dist(xygrid2)
thresh <- 1                    # truncation distance set to 1
# Truncation to threshold 1
xygrid2.d1[xygrid2.d1>thresh] <- 4*thresh
# PCoA of truncated matrix
xygrid2.PCoA <- cmdscale(xygrid2.d1, eig=TRUE,
  k=nrow(xygrid2)-1)
# Count the positive eigenvalues
(nb.ev2 <- length(which(xygrid2.PCoA$eig > 0.0000001)))
# Matrix of PCNM variables
xygrid2.PCNM <- xygrid2.PCoA$points[,1:nb.ev2]

# Plot some PCNM variables modelling positive spatial
# correlation (Fig. 7.4)
par(mfrow=c(4,2))
somePCNM2 <- c(1, 2, 5, 10, 20, 50, 100, 150)
for(i in 1:length(somePCNM2)){
 sr.value(xygrid2, xygrid2.PCNM[,somePCNM2[i]],
 method="greylevel", csize=0.35, sub=somePCNM2[i],
 csub=2)
}
```

Hint *Functions* **`s.value()`** *and* **`sr.value()`** *propose two representations of values on maps: by symbols with sizes proportional to the values (default argument* `method = "squaresize"`*) and by symbols of constant size and values represented by shades of grey (*`method = "greylevel"`*).*

Figures 7.3 and 7.4 show that these variables are periodical and ranging from broadest to finest scales. As discussed above, this does not imply that only periodical structures can be modelled by PCNM analysis, however. Even short-range spatial correlation can be modelled by fine-scale PCNM variables. This topic is addressed later.

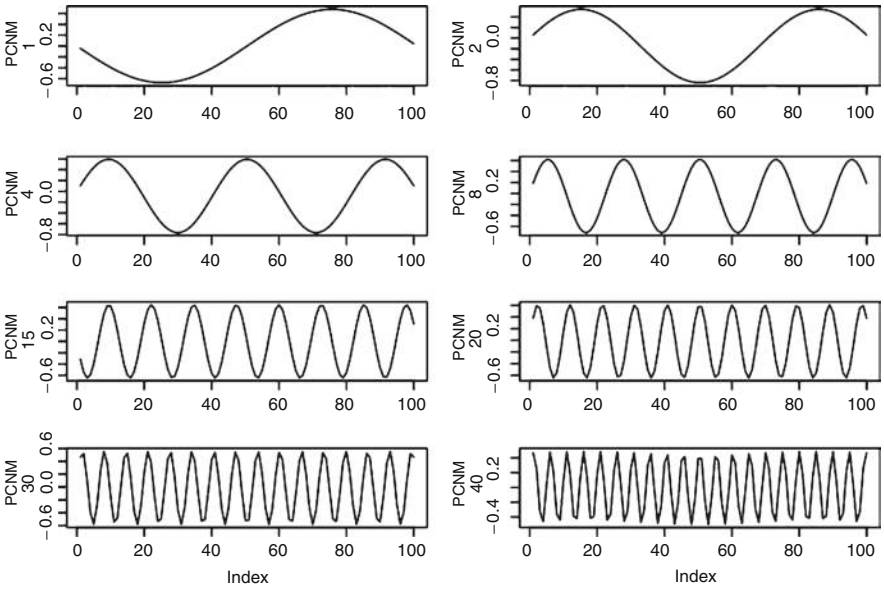

Fig. 7.3 Some of the 67 PCNM variables with positive eigenvalues built from a transect of 100 equispaced points. The first 49 of them have Moran's I larger than $E(I)$, showing that they model positive spatial correlation

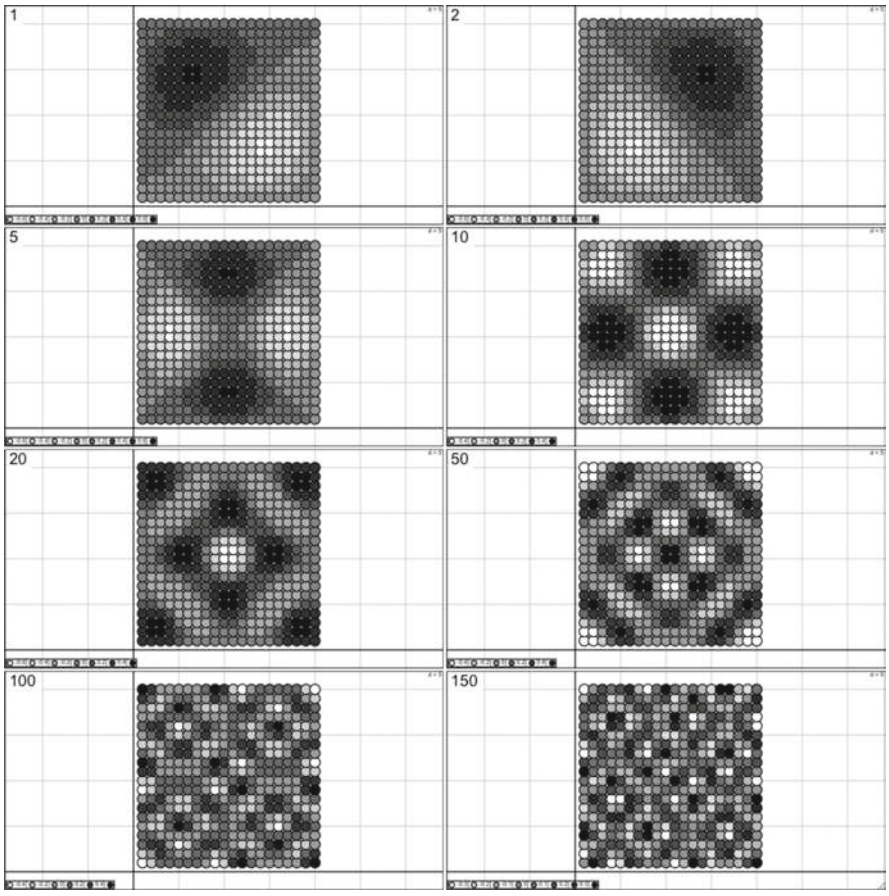

Fig. 7.4 Some of the 279 PCNM variables with positive eigenvalues built from a grid of 20 by 20 equispaced points. The first 209 of them have Moran's I larger than $E(I)$

7.4.2.3 PCNM Analysis of the Mite Data

PCNM analysis is not restricted to regular sampling designs. The drawback in the case of irregular designs is that the PCNM variables lose the regularity of their shapes, sometimes making the assessment of scale more difficult.

Now, it is time to try PCNM analysis on real data. You will first run the analysis "by hand", i.e. by separately coding every step. After that, automated functions will be presented that simplify the analysis.

In the code below, PCNM variables are constructed using a package dedicated to this task (called **PCNM**). Function **PCNM()** of this package provides an immediate assessment of the spatial correlation (Moran's *I*, (7.3)) displayed by the computed eigenfunctions. Moran's *I* gives a criterion to decide which eigenfunctions should be used for modelling; see Sect. 7.4.4. Otherwise, the user can apply the simpler function **pcnm()** of the **vegan** package.

```
# PCNM analysis of the oribatid mite data
# ***************************************

# 1a. Construct the matrix of PCNM variables step by step...
# ----------------------------------------------------------

xy.d1 <- dist(mite.xy)
  # Search for the truncation threshold: maximum value of the
  # minimum spanning tree of the Euclidean distance matrix
  # using vegan's function spanntree. Use that distance, or any
  # other distance larger than that, as the truncation distance.
spanning <- spantree(xy.d1)
dmin <- max(spanning$dist)

# Truncate the distance matrix
xy.d1[xy.d1 > dmin] <- 4*dmin
# PCoA of truncated distance matrix
xy.PCoA <- cmdscale(xy.d1, k=nrow(mite.xy)-1, eig=TRUE)
# Count the positive eigenvalues (PCNM with positive AND
negative spatial
# correlation)
(nb.ev <- length(which(xy.PCoA$eig > 0.0000001)))
# Construct a data frame containing the PCNM variables
mite.PCNM <- as.data.frame(xy.PCoA$points[1:nrow(mite.xy),
1:nb.ev])

# 1b. ... or construct the PCNM variables automatically
# -----------------------------------------------------

# library(PCNM) # If not already loaded
xy.d1 <- dist(mite.xy)
mite.PCNM.auto <- PCNM(xy.d1)
summary(mite.PCNM.auto)
# Plot the minimum spaning tree used to find the truncation
# distance
plot.spantree(mite.PCNM.auto$spanning, mite.xy)
(dmin <- mite.PCNM.auto$thresh) # Truncation distance
(nb.ev <- length(mite.PCNM.auto$values)) # Number of eigenvalues
```

```
# Moran's I of the PCNM variables (in the first distance class,
# 0 to truncation threshold); also see figure generated by the
# PCNM() function (not reproduced here).

# Expected value of I, no spatial correlation
mite.PCNM.auto$expected_Moran
mite.PCNM.auto$Moran_I

# Eigenfunctions with positive spatial correlation
(select <- which(mite.PCNM.auto$Moran_I$Positive == TRUE))
length(select)  # Number of PCNM with I > E(I)
mite.PCNM.pos <- as.data.frame(mite.PCNM.auto$vectors)[,select]

# 1c. ... or use vegan's function pcnm()
# ---------------------------------------
mite.PCNM.vegan <- pcnm(dist(mite.xy))
mite.PCNM <- as.data.frame(mite.PCNM.vegan$vectors)
# dmin <- mite.PCNM.vegan$threshold
nb.ev <- length(which(mite.PCNM.vegan$values > 0.0000001))
 # The eigenvectors obtained by this function are divided by the
 # square root of their eigenvalue. This is not important for
 # their use as spatial variables.
 # Contrary to function PCNM(), vegan's pcnm() does not provide
 # Moran's I,which must be computed separately if one chooses to
 # retain only the eigenfunctions with positive spatial
 # correlation.
```

Hint *The truncation distance can be chosen by the user to be either the value proposed by the PCNM function (longest link along the minimum spanning tree drawn on the map of the points), or any other value larger than that. For example, for a regular two-dimensional grid of points with spacing of 1, one may choose a value slightly larger than the distance between diagonal neighbours, sqrt(2) = 1.4142, as the truncation distance. The chosen truncation distance may be 1.4143 in that case.*

As one can see, the first 17 PCNMs have significant positive spatial correlations at the 5% significance level, while significant negative spatial correlations are found in PCNMs 35–43. The test of significance of Moran's *I* may not be a reliable criterion to eliminate PCNMs from the analysis, however, so we will keep the 23 PCNMs with positive spatial correlation. We will apply forward selection with the Blanchet et al. (2008a) double stopping criterion.

```
# 2. Run the global PCNM analysis on the *detrended* mite data
# ------------------------------------------------------------

mite.PCNM.rda <- rda(mite.h.det, mite.PCNM.pos)
anova.cca(mite.PCNM.rda)

# 3. Since the analysis is significant, compute the adjusted R2
#    and run a forward selection of the PCNM variables

(mite.R2a <- RsquareAdj(mite.PCNM.rda)$adj.r.squared)
(mite.PCNM.fwd <- forward.sel(mite.h.det,
as.matrix(mite.PCNM.pos), adjR2thresh=mite.R2a))
 # According to the R2a criterion, if we retain PCNM 5 we get a
 # model with a R2adj slightly higher than that of the complete
 # model. This slight excess is not too serious, however.

(nb.sig.PCNM <- nrow(mite.PCNM.fwd)) # Number of signif. PCNM

# Identity of significant PCNMs in increasing order
(PCNM.sign <- sort(mite.PCNM.fwd[,2]))
# Write the significant PCNMs to a new object
PCNM.red <- mite.PCNM.pos[,c(PCNM.sign)]

# 4. New PCNM analysis with 10 significant PCNM variables
#    Adjusted R-square after forward selection: R2adj=0.2713

mite.PCNM.rda2 <- rda(mite.h.det ~ ., data=PCNM.red)
(mite.fwd.R2a <- RsquareAdj(mite.PCNM.rda2)$adj.r.squared)
anova.cca(mite.PCNM.rda2)
axes.test <- anova.cca(mite.PCNM.rda2, by="axis")
(nb.ax <- length(which(axes.test[,5] <= 0.05))) # Number of
significant axes

# 5. Plot the two significant canonical axes
mite.PCNM.axes <- scores.cca(mite.PCNM.rda2, choices=c(1,2),
display="lc", scaling=1)
par(mfrow=c(1,2))
sr.value(mite.xy, mite.PCNM.axes[,1]) # ade4 function: s.value
sr.value(mite.xy, mite.PCNM.axes[,2]) # ade4 function: s.value
```

Hint *In the* **`scores.cca()`** *call above, be careful to set* **`display="lc"`**. *The default is* **`"wa"`**, *but here we want the fitted site scores.*

PCNM analysis of the detrended mite data explained 27.13% of the variance (see **`mite.fwd.R2a`**, the adjusted R^2 obtained with the ten variables retained by forward selection, slightly exceeding the total R^2_{adj}). Two canonical axes, explaining $27.13 \times 0.7527^2 = 20.4\%$ of the total variance, are significant; their fitted site

[2] The value 0.7527 is found in the section "Accumulated constrained eigenvalues" of the RDA output. It is the proportion of variance explained by the first two canonical axes with respect to the total *explained* variance.

scores have been plotted on a map of the sites. These two plots (Fig. 7.5) represent the spatially structured variation of the detrended oribatid mite data. Now, is this variation related to any of the environmental variables? A simple way to assess this is to regress the fitted site scores of these two canonical axes on the environmental data.

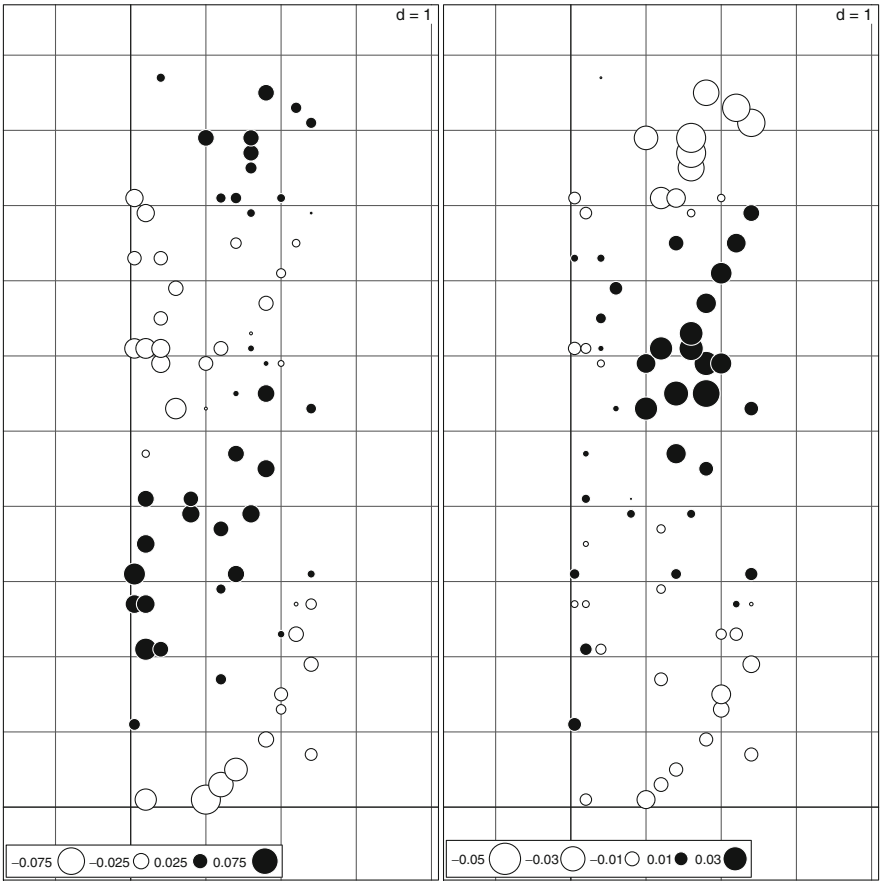

Fig. 7.5 Manual PCNM analysis of the detrended oribatid mite data, ten significant PCNM variables. Maps of the fitted site scores of the first two canonical axes

```
# Interpreting the spatial variation: regression of the two
# significant canonical axes on the environmental variables
# (after normality tests)

shapiro.test(resid(lm(mite.PCNM.axes[,1] ~ ., data=mite.env)))
mite.PCNM.axis1.env <- lm(mite.PCNM.axes[,1] ~ ., data=mite.env)
summary(mite.PCNM.axis1.env)

shapiro.test(resid(lm(mite.PCNM.axes[,2] ~ ., data=mite.env)))
mite.PCNM.axis2.env <- lm(mite.PCNM.axes[,2] ~ ., data=mite.env)
summary(mite.PCNM.axis2.env)

  # As one can see, the two spatial axes are not related to the
  # same environmental variables (except for shrubs). They are
  # not fully explained by them either. A precise assessment of
  # the portions explained will require variation partitioning.
```

This PCNM analysis produced spatial models combining all the PCNM variables forward-selected from the set of 23 classical PCNMs with positive spatial correlation. Here, both significant canonical axes are a combination of PCNM variables ranging from broad (PCNM1) to fine scale (PCNM23). While this may be interesting if one is interested in the global spatial structure of the response data, it does not allow one to discriminate between broad, medium and fine-scale structures since all significant PCNMs are combined.

Another approach consists in computing *separate* RDAs constrained by subsets of the significant PCNM variables. The PCNM variables being linearly independent of one another, any submodel defined with a subset of PCNMs is also independent of any other submodel defined with another subset. These subsets can be defined in such a way as to model different scales. The choices are arbitrary: there is no general rule defining what is broad, medium or fine scale. One can either predefine these limits, using the sizes of the patterns corresponding to the PCNM variables, or run forward selection and define submodels corresponding to groups of more or less consecutive PCNM variables retained by the procedure. One can also draw maps of the significant PCNM variables (Fig. 7.6) and group them according to the scales of the patterns they represent:

```
# Maps of the 10 significant PCNM variables
# ****************************************
par(mfrow=c(2,5))
for(i in 1:ncol(PCNM.red)){
 sr.value(mite.xy, PCNM.red[,i], sub=PCNM.sign[i], csub=2)
}
```

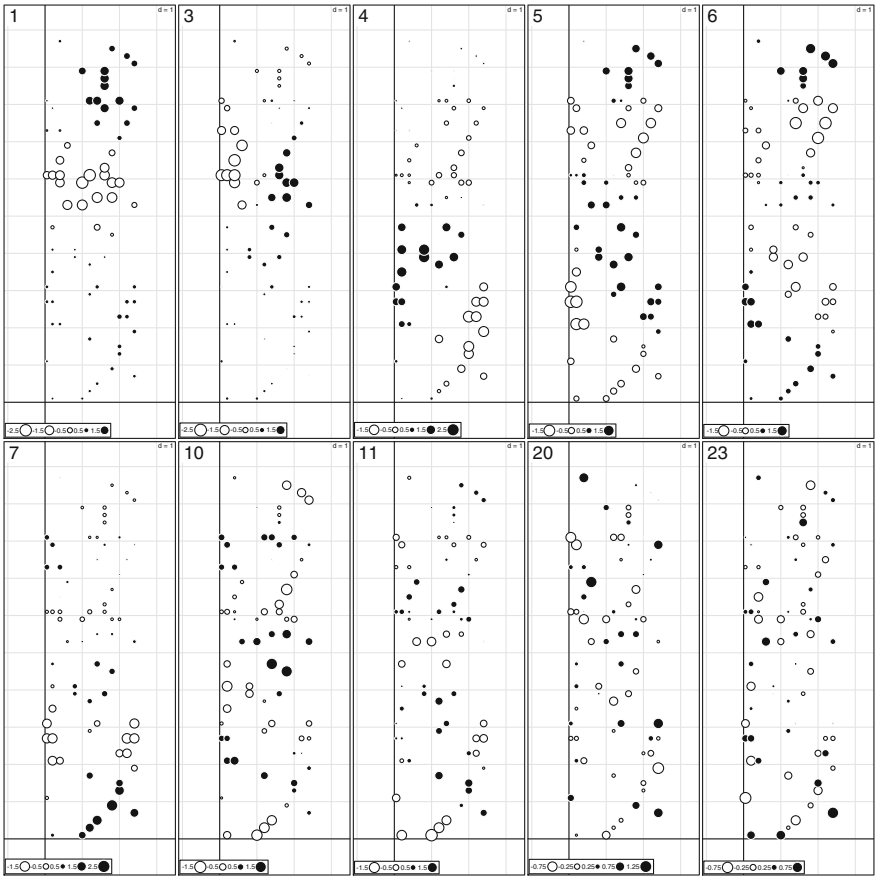

Fig. 7.6 The ten significant PCNM variables with positive spatial correlation used in the manual PCNM analysis of the detrended oribatid mite data

On this basis, one could for instance define PCNMs 1, 3 and 4 as "broad scale", PCNMs 5, 6, 7, 10 and 11 as "medium scale", and PCNMs 20 and 23 as "fine scale" descriptors. Separate RDAs with these subsets model broad, medium and fine scale patterns, respectively.

```
# PCNM analysis of the mite data - broad scale
# ********************************************

mite.PCNM.broad <- rda(mite.h.det ~ .,
 data=mite.PCNM.pos[,c(1,3,4)])
anova.cca(mite.PCNM.broad)
axes.broad <- anova.cca(mite.PCNM.broad, by="axis")
nb.ax.broad <- length(which(axes.broad[,5] <= 0.05))
nb.ax.broad     # Number of significant axes

# Plot of the two significant canonical axes
mite.PCNMbroad.axes <- scores.cca(mite.PCNM.broad,
 choices=c(1,2), display="lc", scaling=1)
par(mfrow=c(1,2))
s.value(mite.xy, mite.PCNMbroad.axes[,1])
s.value(mite.xy, mite.PCNMbroad.axes[,2])

# Interpreting spatial variation: regression of the two
# significant spatial canonical axes on the environmental
# variables

mite.PCNMbroad.ax1.env <- lm(mite.PCNMbroad.axes[,1] ~ .,
 data=mite.env)
summary(mite.PCNMbroad.ax1.env)

mite.PCNMbroad.ax2.env <- lm(mite.PCNMbroad.axes[,2] ~ .,
 data=mite.env)
summary(mite.PCNMbroad.ax2.env)
 # The broad-scale relationships are clearly related to
 # microtopography and the absence of shrubs.


# PCNM analysis of the mite data - medium scale
# *********************************************

mite.PCNM.med <- rda(mite.h.det ~ .,
data=mite.PCNM.pos[,c(5,6,7,10,11)])
anova.cca(mite.PCNM.med)
axes.med <- anova.cca(mite.PCNM.med, by="axis")
nb.ax.med <- length(which(axes.med[,5] <= 0.05))
nb.ax.med      # Number of significant axes

# Plot of the significant canonical axes (the second axis is
# marginally significant)
mite.PCNMmed.axes <- scores.cca(mite.PCNM.med, choices=c(1,2),
 display="lc", scaling=1)
```

```
par(mfrow=c(1,2))
s.value(mite.xy, mite.PCNMmed.axes[,1])
s.value(mite.xy, mite.PCNMmed.axes[,2])

# Interpreting spatial variation: regression of the significant
# spatial canonical axes on the environmental variables

mite.PCNMmed.ax1.env <- lm(mite.PCNMmed.axes[,1] ~ .,
 data=mite.env)
summary(mite.PCNMmed.ax1.env)
mite.PCNMmed.ax2.env <- lm(mite.PCNMmed.axes[,2] ~ .,
 data=mite.env)
summary(mite.PCNMmed.ax2.env)
 # The medium-scale features correspond to several types of soil
 # coverage and to soil moisture (variable WatrCont).


# PCNM analysis of the mite data - fine scale
# ******************************************

mite.PCNM.fine <- rda(mite.h.det ~ .,
data=mite.PCNM.pos[,c(20,23)])
anova.cca(mite.PCNM.fine)
axes.fine <- anova.cca(mite.PCNM.fine, by="axis")
nb.ax.fine <- length(which(axes.fine[,5] <= 0.05))
nb.ax.fine      # Number of significant axes

# Plot of the significant canonical axis
mite.PCNMfine.axes <- scores.cca(mite.PCNM.fine, choices=1,
display="lc", scaling=1)
par(mfrow=c(1,2))
s.value(mite.xy, mite.PCNMfine.axes)

# Interpreting spatial variation: regression of the significant
# spatial canonical axis on the environmental variables

mite.PCNMfine.ax1.env <- lm(mite.PCNMfine.axes ~ .,
data=mite.env)
summary(mite.PCNMfine.ax1.env)
 # The fine-scale structure is weakly related to the
 # environmental variables. Only one soil coverage class (plant
 # litter) is significant.
```

Something has occurred here, which is often found in fine-scale PCNM analysis. The only convincing correlation is with the presence of bare peat, a feature that was very localized in the study area. Otherwise, in most cases, fine scale PCNM

variables cannot be related to environmental descriptors and are mostly the signature of local spatial correlation generated by community dynamics. This topic will be addressed later.

7.4.2.4 Hassle-Free PCNM Analysis: Function `quickPCNM()`

A single-step PCNM analysis can be performed easily with the function **`quickPCNM()`**. This function, available in the **`PCNM`** package, requires only two arguments: a response data table (pre-transformed if necessary) and a table containing the site geographic coordinates (which can be one- or two-dimensional). The function performs a complete PCNM analysis: it checks whether the response data should be detrended and does it if a significant trend is identified; it constructs the PCNM variables and tests the global analysis; it runs forward selection, using the PCNMs with positive spatial correlation; it runs RDA with the retained PCNM variables and tests the canonical axes; it delivers the RDA results (including the set of PCNM variables) and plots maps of the significant canonical axes.

```
# Single-step PCNM analysis using function quickPCNM()
# ****************************************************

mite.PCNM.quick <- quickPCNM(mite.h, mite.xy)
summary(mite.PCNM.quick)
mite.PCNM.quick[[2]]   # Eigenvalues
mite.PCNM.quick[[3]]   # Results of forward selection

  # quickPCNM prevents forward.sel() from admitting a variable
  # when the resulting R2adj exceeds that of the complete model.
  # Therefore, PCNM5 is not among the selected PCNMs in the
  # quickPCNM results.
```

Function **`quickPCNM()`** provides several arguments to fit various needs. For instance, detrending is done by default if a significant trend is found, but this option can be disabled (`detrend=FALSE`). The truncation threshold is computed automatically unless the user provides another value (e.g. `thresh=1.234`). Computation of the PCNM variables is overridden if the user provides a ready-made set of spatial regressors (`myPCNM=userdataset`).

`quickPCNM()` provides a composite output object containing many results. The summary shows all the components. To draw a biplot of the RDA results, the code is the following:

```
# Extract and plot RDA results from a quickPCNM output
# (scaling 2)
# ************************************************

plot(mite.PCNM.quick$RDA, scaling=2)
sp.scores2 <- scores(mite.PCNM.quick$RDA,  choices=1:2,
  scaling=2, display="sp")
arrows(0, 0, sp.scores1[,1], sp.scores1[,2], length=0, lty=1
   col="red")

  # The scaling 2 shows the relationship of some species with
  # some PCNM variables. These correlations can be explored to
  # reveal at which scale the species distributions are spatially
  # structured.
```

7.4.2.5 Combining PCNM Analysis and Variation Partitioning

A clever and global approach to assess the environmental variation related to all scales of spatial variation is to perform a *variation partitioning* with an environmental data set and up to three subsets of spatial variables. Function **`varpart()`** can only handle numeric variables (not factors), however, so that we have to recode environmental variables 3–5 into dummy binary variables.

Variation partitioning aims at quantifying the various unique and combined fractions of variation explained by several sources. In this context, a linear trend can be considered as a source of variation like any other. The trend is likely to act on the response as well as the explanatory variables. Therefore, in this application we advocate *not* to detrend the response data prior to variation partitioning, but rather to test for a linear trend and incorporate it explicitly in the partitioning procedure if it is significant.

In this example, we independently forward-select the *X*–*Y* coordinates, the environmental variables and the PCNM variables before variation partitioning. The significant PCNM variables are split into a broad and a fine-scale fraction. The partitioning results are presented in Fig. 7.7.

```
# Mite - trend - environment - PCNM variation partitioning
# ********************************************************

# 1. Test trend. If significant, forward selection of
#    coordinates
# -----------------------------------------------------

mite.XY.rda <- rda(mite.h, mite.xy)
anova.cca(mite.XY.rda)
(mite.XY.R2a <- RsquareAdj(mite.XY.rda)$adj.r.squared)
(mite.XY.fwd <- forward.sel(mite.h, as.matrix(mite.xy),
  adjR2thresh=mite.XY.R2a))
XY.sign <- sort(mite.XY.fwd$order)
# Write the significant coordinates to a new object
XY.red <- mite.xy[,c(XY.sign)]

# 2. Test and forward selection of environmental variables
# -------------------------------------------------------

# Recode environmental variables 3 to 5 into dummy binary
# variables
substrate <- model.matrix(~mite.env[,3])[,-1]
shrubs <- model.matrix(~mite.env[,4])[,-1]
topo <- model.matrix(~mite.env[,5])[,-1]
mite.env2 <- cbind(mite.env[,1:2], substrate, shrubs, topo)

# Forward selection of the environmental variables
mite.env.rda <- rda(mite.h, mite.env2)
(mite.env.R2a <- RsquareAdj(mite.env.rda)$adj.r.squared)
mite.env.fwd <- forward.sel(mite.h, mite.env2,
adjR2thresh=mite.env.R2a, nperm=9999)
env.sign <- sort(mite.env.fwd$order)
env.red <- mite.env2[,c(env.sign)]
colnames(env.red)

# 3. Test and forward selection of PCNM variables
# -----------------------------------------------

# Run the global PCNM analysis on the *undetrended* mite data
mite.undet.PCNM.rda <- rda(mite.h, mite.PCNM.pos)
anova.cca(mite.undet.PCNM.rda)
# Since the analysis is significant, compute the adjusted R2
#    and run a forward selection of the PCNM variables
(mite.undet.PCNM.R2a <-
 RsquareAdj(mite.undet.PCNM.rda)$adj.r.squared)
(mite.undet.PCNM.fwd <- forward.sel(mite.h,
```

```
  as.matrix(mite.PCNM.pos),
  adjR2thresh=mite.undet.PCNM.R2a))

  # According to the R2a criterion, if we retain 12 PCNMs
  # we get a model with a R2adj slightly higher than that of the
  # complete model. This slight excess is not too serious,
  # however.

# Number of signif. PCNM
(nb.sig.PCNM <- nrow(mite.undet.PCNM.fwd))
# Identity of significant PCNMs in increasing order
(PCNM.sign <- sort(mite.undet.PCNM.fwd$order))
# Write the significant PCNMs to a new object
PCNM.red <- mite.PCNM.pos[,c(PCNM.sign)]

# 4. Arbitrary split of the significant PCNMs into broad and
# fine scale
# ------------------------------------------------------------
# Broad scale: PCNMs 1, 2, 3, 4, 6, 7, 8, 9, 10, 11
PCNM.broad <- PCNM.red[,1:10]
# Fine scale: PCNMs 16, 20
PCNM.fine <- PCNM.red[,11:12]

# 5. Mite - environment - trend - PCNM variation partitioning
# ------------------------------------------------------------

(mite.varpart <- varpart(mite.h, env.red, XY.red, PCNM.broad,
  PCNM.fine))
par(mfrow=c(1,2))
showvarparts(4)
plot(mite.varpart, digits=2)
# The default options leave the fractions with negative R2a
# empty. To display the negatives values, set the argument
# cutoff = -Inf.

# Tests of the unique fractions [a], [b], [c] and [d]
# ***************************************************
# Fraction [a], pure environmental
anova.cca(rda(mite.h, env.red, cbind(XY.red, PCNM.broad,
  PCNM.fine)))
# Fraction [b], pure trend
anova.cca(rda(mite.h, XY.red, cbind(env.red, PCNM.broad,
  PCNM.fine)))
# Fraction [c], pure broad scale spatial
anova.cca(rda(mite.h, PCNM.broad, cbind(env.red, XY.red,
   PCNM.fine)))
```

```
# Fraction [d], pure fine scale spatial
anova.cca(rda(mite.h, PCNM.fine, cbind(env.red, XY.red,
  PCNM.broad)))
 # Only the pure environmental and the pure broad scale
 # spatial fractions are significant.
```

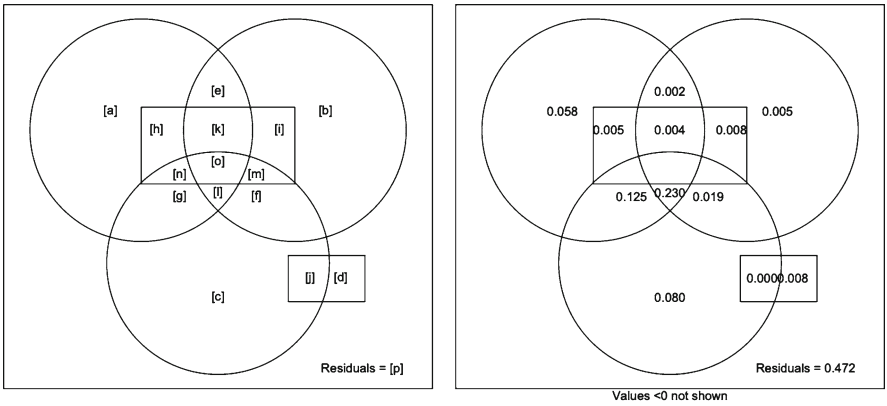

Fig. 7.7 Variation partitioning of the undetrended oribatid mite data into an environmental component (*upper left-hand circle*), a linear trend (*upper right-hand circle*), a broad scale (*lower circle*) and fine scale (*disjoined rectangles*) PCNM spatial components. The empty fractions in the plots have small negative R^2_{adj} values

When interpreting such a complex variation partitioning diagram, keep in mind that the R^2 adjustment is done for each fraction that can be fitted without resorting to partial RDA or multiple regression (here, the first 15 rows of the table of results), and that the individual fractions [a] to [p] are then computed by subtraction. Very small negative R^2_{adj} values frequently appear in this process. Negative R^2_{adj} values correspond to explanatory variables that explain less of the response variables' variation than would be expected by chance; so, for all practical purposes, they can be interpreted as zeros and neglected during interpretation, although they must be taken into account for the sum of all fractions to be 1.

The whole set of environmental and spatial variables explains 52.8% of the variation of the undetrended mite data (see the R^2_{adj} for "All" fractions). The environmental variables alone (matrix **X**1 in the partitioning results) explain 40.8% of the variation, of which a mere 5.8% is not spatially structured (fraction [a]). This fraction represents species–environment relationships associated with local environmental conditions.

The remaining fractions involving environmental and spatial variables (essentially fractions [g] and [l]) represent spatially structured environmental variation. Fraction [g] (12.5% variance explained) is common to the environmental and broad scale PCNM variables. Fraction [l] (23.0%) represents a strong spatial component that is jointly explained by the environmental variables, the *Y* coordinate of the sampling sites and the broad scale PCNM variation. This is a typical case of *induced spatial variation*, where the spatial structure of environmental factors produces a similar spatial structure in the response data. In this example, fraction [l], which represents two-thirds of that joint structure, corresponds to the linear gradient in the north–south direction of the map represented in the analysis by variable mite.xy[,2], showing that broad-scale PCNM variables can indeed model a linear gradient. On the other hand, the common fractions corresponding to the environment and the fine-scale PCNM structure ([h + k + n + o], $R^2_{\text{adj}} = -0.006\%$) is negligible.

When some variance is explained commonly by the environmental and spatial variables, one should be careful when inferring causal species–environment relationships: the correlations may be due to a direct influence of the environmental variables on the species (direct induced spatial variation), or to some unmeasured underlying process that is spatially structured and is influencing both the mite community and the environmental variables (e.g. spatial variation induced by a historical causal factor).

The variation partitioning also shows that the four sources of variation have unequal unique contributions: the environment alone ([a], 5.8%), as well as the broad scale ([c], 8.0%) variation, are significant, while the trend alone ([b], 0.5%) and the fine scale variation ([d], 0.8%) are not.

There is also some variation explained by spatial variables independently of the environment. This variation is represented by fractions [b], [c], [d], [f], [i], [j] and [m]. Together these fractions explain 12% of the variation. Most likely, some of this variation, especially at broad and medium scales, could be explained by unmeasured environmental variables, although one cannot exclude the influence of past events that could still show their marks in the mite community (Borcard and Legendre 1994). Fine scale structures are more likely explainable by spatial correlation produced by neutral biotic processes. Neutral processes include ecological drift (variation in species demography due to random reproduction and random survival of individuals due to competition, predator–prey interactions, etc.) and random dispersal (migration in animals, propagule dispersion in plants). Controlling for spatial correlation by means of PCNM variables when testing species–environment relationships is briefly addressed in Sect. 7.4.3.4.

Finally, note that the broad and fine-scale PCNM variables have a non-null intersection despite the fact that the PCNM variables are orthogonal: fraction [j+m+n+o] totals −1.7%. This occurs because other variables (environment and trend), which are not orthogonal to the PCNMs, are involved in the partitioning, and also because the variation partitioning procedure involves subtractions of R^2 that have been adjusted on the basis of different numbers of explanatory variables.

7.4.3 MEM in a Wider Context: Weights Other than Geographic Distances

7.4.3.1 Introduction

The PCNM method provides an elegant way of constructing sets of linearly independent spatial variables. Since its publication, it has gained a wide audience and has been applied in several research papers. But it is not the end of the story.

Dray et al. (2006) have greatly improved the mathematical formalism of PCNM analysis by showing that it is a particular case of a wider family of methods that they called *Moran's eigenvector maps* (MEM). They demonstrated the link between the eigenvalues of the MEM eigenvectors and Moran's spatial correlation index, *I* (7.3).

They reasoned that the relationship among sites, which is the basis for any spatial eigenvector decomposition, actually has two components: (1) a list of *links* among objects, represented by a *connectivity matrix* and (2) a matrix of *weights* to be applied to these links. In the simplest case, the weights are binary (i.e. either two objects are linked, or they are not). In more complex models, non-negative weights can be placed on the links; these weights represent the easiness of exchange (of organisms, energy, information, etc.) between the points connected by the links. For instance, link weights can be made to be inversely proportional to the squared Euclidean distance among sites.

Furthermore, Dray et al. (2006) showed that (1) by using similarities instead of distances among sites, (2) setting the relationship of the sites with themselves to null similarity and (3) avoiding a square-root standardization of the eigenvectors within the PCoA procedure, one obtains a family of flexible methods (MEM) that bear an immediate connexion with Moran's *I* and can be modulated to optimize the construction of spatial variables. The MEM method produces $n-1$ spatial variables with positive *and* negative eigenvalues, allowing the construction of a wide range of variables modelling positive and negative spatial correlation. The eigenvectors maximize Moran's *I* index, the eigenvalues being equal to Moran's *I* multiplied by a constant. Therefore, the spatial structures of the data are extracted in such a way

that the axes first optimally display the positively autocorrelated structures in decreasing order of importance, and then the negatively autocorrelated structures in increasing order.

The MEM method consists in defining two matrices describing the relationships among the sites:

- A binary **connectivity** matrix **B** defining which pairs of sites are connected (1) and which are not (0)
- A **weighting** matrix **A** providing the intensity of the connexions

The final *spatial weighting matrix* **W** results from the Hadamard (i.e. term-by-term) product of these two matrices, **B** and **A**.

The connectivity matrix **B** can be constructed on the basis of distances (by selecting a distance threshold and connecting all points that are within that distance) or by other connexion schemes, such as Delaunay triangulation, Gabriel graph or others (described by Legendre and Legendre 1998, Section 13.3). The connexion matrix can of course be customized to fit special needs – for instance, by only allowing connexions among sites along the littoral zone of a lake (not across water) or along the shoreline of an island.

Matrix **A** is not mandatory, but is often used to weight the connexions according to distance, e.g. by inverse distance or inverse squared distance, since it is ecologically realistic to assume that a process influences a community with an intensity decreasing with distance. The choice of both matrices is very important because it greatly affects the structure of the spatial variables obtained. These variables, in turn, condition the results of the spatial analysis, especially in the case of irregular sampling: "*In the case of **regular** sampling (e.g. a regular grid), structures defined by eigenvectors are roughly similar for different definitions of **W**. For **irregular** distributions of sites, however, the number of positive/negative eigenvalues and the spatial structures described by their associated eigenvectors are greatly influenced by the spatial relationships defined in **W***" (Dray et al. 2006). These authors provide the following general recommendations:

> *The choice of the spatial weighting matrix **W** is the most critical step in spatial analysis. This matrix is a model of the spatial interactions recognized among the sites, all other interactions being excluded. In some cases, a theory-driven specification can be adopted, and the spatial weighting matrix can be constructed based upon biological considerations […]. In most situations, however, the choice of a particular matrix may become rather difficult and a data-driven specification could then be applied. Under this latter approach, the objective is to select a configuration of **W** that results in the optimal performance of the spatial model.*

For data-driven model specification, the authors proposed a procedure starting with a user-defined set of possible spatial weighting matrices. For *each* candidate, one computes the MEM eigenfunctions, reorders them according to their explanatory power, enters them one by one into the model and retains the model with the lowest corrected Akaike information criterion (AIC_c). When this is done for all candidates, one retains the **W** matrix yielding the lowest AIC_c.

The AIC_c-based selection is but one possibility. One could also forward-select the MEM within each candidate model using Blanchet et al.'s (2008a) double stopping criterion and retain the model with the highest R^2_{adj}. This alternative, which had not yet been devised when the Dray et al. (2006) paper was published, addresses the concerns raised by these authors in their conclusion about the drawbacks of forward selection procedures.

7.4.3.2 MEM Analysis of the Mite Data

Dray et al. (2006) used the oribatid mite data to illustrate MEM analysis. As an example, we duplicate their analysis, exploring some choices along the steps of the method. Several packages are used. The following example is based on Stéphane Dray's tutorial on MEM analysis, with our thanks to the author. It relies heavily on the **`spacemakeR`** package that Dray devised especially for this purpose.

The **`spacemakeR`** functions used below should make model selection relatively easy. Of course, the final result depends upon a proper choice of a class of models. The function **`test.W()`** is particularly useful as it combines the construction of MEM variables and model selection; examine the help file of that function.

We experiment with three classes of models:

- The first class is based on Delaunay triangulation with binary weights.
- The second class starts from the same connectivity matrix, to which weights are added. The weighting function is based on Euclidean distances among the sites: $f_2 = 1 - (d/d_{max})^{\alpha}$ where d is a distance value and d_{max} is the maximum value in the distance matrix.
- The third class evaluates a series of models based on a range of distances around the points. All pairs of points within the distance considered are linked, the others not. What should the range of distances be? This can be assessed by means of a multivariate variogram of the response data. Variograms are plots of semivariances against distance classes. Semivariance is a distance-dependent measure of variation, which is used in the same way as in the correlograms presented earlier (e.g. Bivand et al. 2008). A variant of this approach will weight the links by the

function of inverse distance that was used in the second model class above. This last variant duplicates the results presented in the Dray et al. (2006) paper.

First and second classes of MEM models: unweighted (binary) and distance-weighted Delaunay triangulation.

```
# MEM analysis of the detrended oribatid mite data
# ************************************************

# Selection of an optimal spatial weighting matrix
# ************************************************

# 1. Search based on Delaunay triangulation.
#    We use mite.h.det as response data and mite.del as Delaunay
#    triangulation data.
#    No weighting matrix (binary weights); function test.W
#    selects among the MEM variables constructed on the basis of
#    the Delaunay triangulation.

# Delaunay triangulation
(mite.del <- tri2nb(mite.xy))
mite.del.res <- test.W(mite.h.det, mite.del)

  # The screen output says that the best model has an AICc value
  # of -94.2 and is based on 7 MEM variables.

# Summary of the results for the best model
summary(mite.del.res$best)

# Unadjusted R^2 of best model
# This line returns the R^2 of the model with the smallest AICc
(R2.del <-
 mite.del.res$best$R2[which.min(mite.del.res$best$AICc)])

# Adjusted R^2 of best model (n = 70 and m = 7)
RsquareAdj(R2.del,70,7)


# 2. Delaunay triangulation weighted by a function of distance.
#    Distances are ranged to maximum 1, and raised to power
#    alpha

f2 <- function(D, dmax, y) { 1 - (D/dmax)^y }

# Largest Euclidean distance on links belonging to the Delaunay
# triangulation
```

```
max.d1 <- max(unlist(nbdists(mite.del, as.matrix(mite.xy))))
# Power is set from 2 to 10
mite.del.f2 <- test.W(mite.h.det, mite.del, f=f2, y=2:10,
dmax=max.d1, xy=as.matrix(mite.xy))

  # The screen output says that the best model has an AICc value
  # of -95.4 and is based on 6 MEM variables.

# Unadjusted R^2 of best model
(R2.delW <-
  mite.del.f2$best$R2[which.min(mite.del.f2$best$AICc)])
# Adjusted R^2 of best model (n = 70 and m = 6)
RsquareAdj(R2.delW,70,6)
```

Third class of MEM models: connectivity matrix based on distances.

```
# 3a. Connectivity matrix based on a distance (radius around
#     points)

# Assessment of the relevant distances, based on a multivariate
# variogram of the detrended mite data, with 20 distance
# classes.

(mite.vario <- variogmultiv(mite.h.det, mite.xy, nclass=20))
plot(mite.vario$d, mite.vario$var, ty='b', pch=20,
  xlab="Distance", ylab="C(distance)")
```

The multivariate variogram is presented in Fig. 7.8. It consists in the sum of univariate variograms computed for all species. The variance increases from 0 to 4 m. Since the shortest distance to keep all sites connected is 1.0111 m (see PCNM analysis), we explore a range of ten evenly distributed distances ranging from this threshold up to 4.0 m (i.e. approximately four times the threshold, as in PCNM analysis).

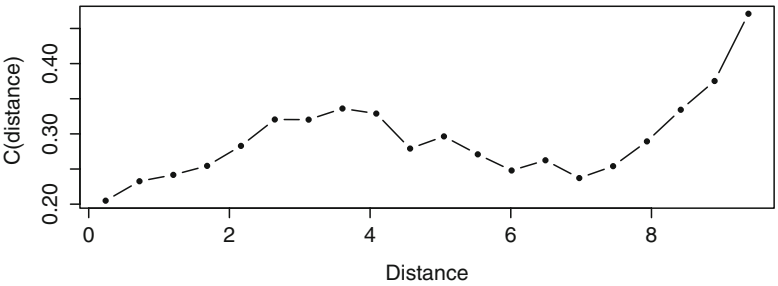

Fig. 7.8 Multivariate variogram of the detrended oribatid mite data. Twenty distance classes

```
# Construction of 10 neighbourhood matrices (class nb)
# Vector of 10 threshold distances
(thresh10 <- seq(give.thresh(dist(mite.xy)), 4, le=10))

# Create 10 neighbourhood matrices.
# Each matrix contains all connexions with lengths ≤ the
# threshold value
list10nb <- lapply(thresh10, dnearneigh, x=as.matrix(mite.xy),
 d1=0)

# Display an excerpt of the first neighbourhood matrix
print(listw2mat(nb2listw(list10nb[[1]], style="B"))[1:10,1:10],
 digits=1)

# Now we can apply the function test.W() to the 10 neighbourhood
# matrices. There are no weights on the links.
mite.thresh.res <- lapply(list10nb, test.W, Y=mite.h.det)

# Lowest AICc, best model, threshold distance of best model
mite.thresh.minAIC <- sapply(mite.thresh.res, function(x)
min(x$best$AICc, na.rm=TRUE))
# Smallest AICc (best model among the 10)
min(mite.thresh.minAIC)
# Number of the model among the 10
which.min(mite.thresh.minAIC)
# Truncation threshold (distance)
thresh10[which.min(mite.thresh.minAIC)]
 # Identify the best model, that is, the model with the lowest
```

```
 # AICc. What is the range of distances in that model? How many
 # MEMs were selected?
```

Hint **dnearneigh()** *requires two geographic dimensions. Add a constant column (e.g. a column of 1) if you only have one dimension, e.g. a transect or a time series.*

This result is more interesting than that of the weighted Delaunay MEM. The AICc of the best model, obtained with a threshold of 2 m, is −100.6 with a model consisting of five MEM variables only. Let us see if we could improve this result by weighting the connexions by an inverse distance function.

```
# 3b. Variant: same as above, but connections weighted by the
#     complement of the power of the distances, 1-(d/dmax)^y

mite.thresh.f2 <- lapply(list10nb, function(x) test.W(x,
  Y=mite.h.det, f=f2, y=2:10, dmax=max(unlist(nbdists(x,
  as.matrix(mite.xy)))), xy=as.matrix(mite.xy)))

# Lowest AIC, best model
mite.f2.minAIC <- sapply(mite.thresh.f2, function(x)
min(x$best$AICc, na.rm=TRUE))

# Smallest AICc (best model among the 10
min(mite.f2.minAIC))

# Number of the model among the 10
(nb.bestmod <- which.min(mite.f2.minAIC))

# Actual dmax of best model
(dmax.best <- mite.thresh.f2[nb.bestmod][[1]]$all[1,2])
```

Hint *The actual d_{max} value found by the function is often smaller than the d_{max} provided to the function by means of the vector of user-selected threshold distances, because the output of the function shows the largest actual distance within the limit provided by each threshold value. In the example, the sixth value in vector* `thresh10`*, which contains the list of user-selected threshold distances, is 2.671639. There is no such distance in the mite geographic distance matrix; the function found that the largest distance smaller than or equal to that threshold is 2.668333.*

With an AIC_c of −102.7, this is the best result of all our attempts in terms of AIC_c. We can therefore extract this champion model, which contains seven MEM variables, from the output object:

```
# Extraction of the champion MEM model
# ************************************
mite.MEM.champ <-
 unlist(mite.thresh.f2[which.min(mite.f2.minAIC)],
 recursive=FALSE)
summary(mite.MEM.champ)

mite.MEM.champ$best$values  # Eigenvalues
mite.MEM.champ$best$ord     # MEM variables by order of added R2

# MEM variables selected in the best model
MEMid <-
 mite.MEM.champ$best$ord[1:which.min(mite.MEM.champ$best$AICc)]
sort(MEMid)
MEM.all <- mite.MEM.champ$best$vectors
MEM.select <- mite.MEM.champ$best$vectors[, sort(c(MEMid))]
colnames(MEM.select) <- sort(MEMid)
# Unadjusted R2 of best model
R2.MEMbest <-
 mite.MEM.champ$best$R2[which.min(mite.MEM.champ$best$AICc)]
# Adjusted R2 of best model
RsquareAdj(R2.MEMbest, nrow(mite.h.det), length(MEMid))

# Plot the links using the function plot.links()
plot.links(mite.xy, thresh=dmax.best)

# Maps of the 7 significant MEM variables
# ***************************************
par(mfrow=c(2,4))
for(i in 1:ncol(MEM.select)){
s.value(mite.xy,MEM.select[,i], sub=sort(MEMid)[i], csub=2)
}
```

The very best MEM model among those tested contains seven MEM variables; four of them are positively spatially correlated (1, 2, 3, 6) and three negatively (9, 11, 57). Interestingly enough, the same result is found by redoing the selection using the forward selection procedure proposed by Blanchet et al. (2008a), with two separate forward selections for the MEM with positive and negative spatial correlation and using only the α value as stopping criterion for the negative MEM.

RDA of the detrended mite data with the seven MEM variables can be computed in a similar fashion as in the PCNM analysis:

```
# RDA of the mite data constrained by the 7 MEM retained, using
# vegan
# ************************************************************
(mite.MEM.rda <- rda(mite.h.det~., as.data.frame(MEM.select)))
(mite.MEM.R2a <- RsquareAdj(mite.MEM.rda)$adj.r.squared)
anova.cca(mite.MEM.rda)
axes.MEM.test <- anova.cca(mite.MEM.rda, by="axis")
# Number of significant axes
(nb.ax <- length(which(axes.MEM.test[,5] <= 0.05)))

# Plot maps of the two significant canonical axes
mite.MEM.axes <- scores.cca(mite.MEM.rda, choices=c(1,2),
 display="lc", scaling=1)
par(mfrow=c(1,2))
s.value(mite.xy, mite.MEM.axes[,1])
s.value(mite.xy, mite.MEM.axes[,2])
```

The R^2_{adj} of the MEM and PCNM models are similar (approximately 0.30), but the PCNM model requires 11 variables to reach this value and is thus less parsimonious than the MEM model. The graphical result (not reproduced here) closely resembles that of the PCNM analysis, showing that the structures revealed by the two analyses are the same.

For the sake of comparison with the PCNM variables, one can plot the seven MEM variables on a map of the sampling area:

```
# Maps of the 7 significant MEM variables
# ***************************************

par(mfrow=c(2,4))
for(i in 1:ncol(MEM.select)){
s.value(mite.xy, MEM.select[,i], sub=sort(MEMid)[i], csub=2)
}
```

The MEMs look similar to the PCNMs at first glance, but a closer look shows that they differ more than one would have expected. Indeed, the two groups of spatial variables are rather weakly correlated:

```
# Correlation of the retained MEM and PCNM variables
# --------------------------------------------------
cor(MEM.select, PCNM.red)
```

These MEM results show that the whole process of selecting and fine-tuning a spatial model, cumbersome as it may seem, can end up with an efficient and parsimonious set of spatial variables.

Note also that **`test.W()`** searches the best model, including MEMs with positive *and* negative spatial correlation. To identify the two groups of eigenfunctions and, if desired, run separate analyses, one must compute Moran's *I* of the selected MEMs found in object `MEM.select`. Another approach could be to compute Moran's *I* of all MEM candidates of the champion model (object `MEM.all`) and run separate forward selections on MEMs with positive and negative spatial correlation.

7.4.3.3 Other Types of Connectivity Matrices

In special cases, when one has a specific spatial model in mind, it is useless to go through the automatic procedure shown above, which finds the best model among multiple possibilities. The present subsection shows how to construct connectivity matrices of several types by hand.

Apart from the Delaunay triangulation used in the example above, the package **`spdep`** offers many possibilities for the definition of connectivity matrices. The ones constructed below and shown in Fig. 7.9 are described in Legendre and Legendre (1998), Section 13.3. They are presented in decreasing order of connectivity and are nested (i.e. the edges (connexions) of a minimum spanning tree are all included in the relative neighbourhood graph and so on).

Depending on the context (hypotheses, data), researchers need connecting schemes that are more or less dense. Some reasons may be technical (e.g. the use of a minimum spanning tree in the PCNM procedure presented in Sect. 7.4.2). Ecological reasons include topographical structure of the sampling area (including possible barriers), dispersion ability of organisms, permeability of some types of substrates and so on.

```
# Connectivity matrices in decreasing order of connectivity
# Delaunay triangulation (as in the previous example)
mite.del <- tri2nb(mite.xy)
# Gabriel graph
mite.gab <- graph2nb(gabrielneigh(as.matrix(mite.xy)), sym=TRUE)
# Relative neighbourhood
mite.rel <- graph2nb(relativeneigh(as.matrix(mite.xy)),
  sym=TRUE)
# Minimum spanning tree
mite.mst <- mst.nb(dist(mite.xy))
  # All these neighbourhood matrices are stored in objects of
  # class nb.
```

```
# Plots of the connectivity matrices
par(mfrow=c(2,2))
plot(mite.del, mite.xy, col="red", pch=20, cex=1)
title(main="Delaunay triangulation ")
plot(mite.gab, mite.xy, col="purple", pch=20, cex=1)
title(main="Gabriel graph")
plot(mite.rel, mite.xy, col="dark green", pch=20, cex=1)
title(main="Relative neighbourhood")
plot(mite.mst, mite.xy, col="brown", pch=20, cex=1)
title(main="Minimum spanning tree")
```

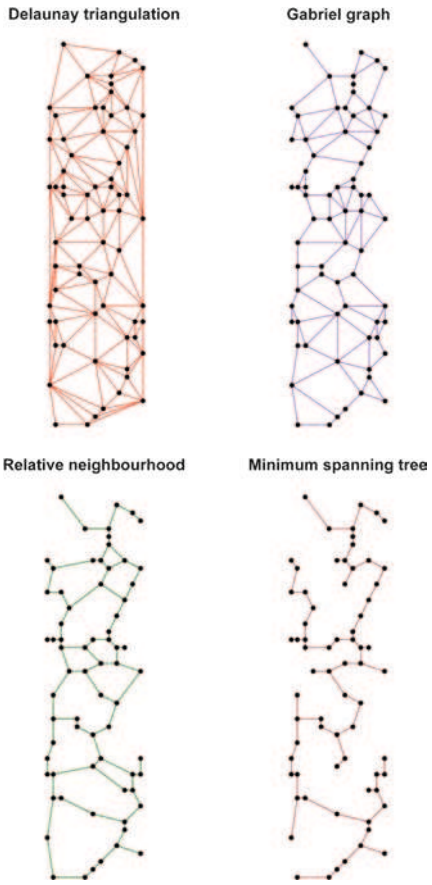

Fig. 7.9 Four types of connectivity matrices applied to the oribatid mite sampling plot. They are presented in decreasing order of connectivity. The links in each model are a subset of the links in the previous one

Some of these matrices may contain unwanted links (for instance, along the borders of the areas). These can be edited either interactively or by command lines:

```
# Link editing
# ************

# 1. Interactive:
plot(mite.del, mite.xy, col="red", pch=20, cex=2)
title(main="Delaunay triangulation ")

mite.del2 <- edit.nb(mite.del,mite.xy)

  # To delete a link, click on its two nodes. Follow on-screen
  # instructions.
  # Wait until you have finished editing before entering the next
  # command line.

# 2. Alternately, links can also be removed by command lines,
# after having converted the nb object into an editable matrix:

mite.del.mat <- nb2mat(mite.del, style="B")
# Remove connection between objects 23 and 35:
mite.del.mat[23,35] <- 0
mite.del.mat[35,23] <- 0
# Back-conversion into nb object:
mite.del3 <- neig2nb(neig(mat01=mite.del.mat))

plot(mite.del3, mite.xy)

# Example: list of neighbours of site 23 for the Delaunay
triangulation:
mite.del[[23]]   # Before editing
mite.del2[[23]]  # After interactive editing
mite.del3[[23]]  # After command line editing
```

The following code shows how to construct connectivity matrices based on a distance: pairs of sites within a given radius are connected, the others not.

```
# Connectivity matrix based on a distance (radius around points)
# Using the same truncation distance dmin as in the PCNM
# example: dmin = 1.011187

mite.thresh4 <- dnearneigh(as.matrix(mite.xy), 0, dmin*4)
nb2mat(mite.thresh4)[1:10, 1:10] # Display of some values
```

```
# Using a shorter distance (1*dmin, 2*dmin)

mite.thresh1 <- dnearneigh(as.matrix(mite.xy), 0, dmin*1)
mite.thresh2 <- dnearneigh(as.matrix(mite.xy), 0, dmin*2)

# Using a longer distance
mite.thresh8 <- dnearneigh(as.matrix(mite.xy), 0, dmin*8)

# Plot of some connectivity matrices
par(mfrow=c(1,2))
plot(mite.thresh1, mite.xy, col="red", pch=20, cex=0.8)
title(main="1 * dmin")
plot(mite.thresh4, mite.xy, col="red", pch=20, cex=0.8)
title(main="4 * dmin")

  # The 1*dmin version shows one disconnected point (7). To avoid
  # such problems, use a slightly larger dmin value. In this case
  # 1.0011188 is enough. The 4*dmin version is very crowded. A
  # lot of links are possible within a little more than 4 meters
  # around each point.
```

These connectivity matrices belong to class "nb". To use them further, we need to convert them into another class called "listw". The function doing this conversion is called **nb2listw()**.

In the simplest case, one of the binary matrices above can be directly converted as follows (including a matrix-class representation of the connectivity matrix for convenience, using function **listw2mat())**:

```
# Conversion of a "nb" object into a "listw" object
# Example: mite.thresh4 created above. "B" is for "binary"

mite.thresh4.lw <- nb2listw(mite.thresh4, style="B")
print(listw2mat(mite.thresh4.lw)[1:10, 1:10], digits=1)
```

This binary (unweighted) matrix could be used directly to create MEM variables using the function **scores.listw()**; see below.

Now, if you want to apply weights (matrix **A**) onto a binary matrix on the basis of Euclidean distances, you need two additional steps: (1) replace all values "1" in the connectivity matrix by the corresponding Euclidean distances [function **nbdists()**] and (2) define weights as a function of inverse distances in this example (weights may be different in other examples):

```
# Creation of a spatial weighting matrix W = Hadamard product of
# B and A
# ------------------------------------------------------------

# Replace "1" by Euclidean distances in the connectivity matrix
mite.thresh4.d1 <- nbdists(mite.thresh4, as.matrix(mite.xy))

# Weights as function of inverse distance
mite.inv.dist <- lapply(mite.thresh4.d1, function(x) 1-
x/max(dist(mite.xy)))

# Creation of spatial weighting matrix W. Argument "B" stands
# for "binary" but concerns the links themselves, not their
# weights
mite.invdist.lw <- nb2listw(mite.thresh4, glist=mite.inv.dist,
  style="B")
print(listw2mat(mite.invdist.lw)[1:10, 1:10], digits=2)

  # All nonzero links have been replaced by weights proportional
  # to the inverses of the Euclidean distances between the
  # points.
```

Now, it is time to compute the MEM spatial variables. This can be done by function **`scores.listw()`** of the package **`spacemakeR`**. We do it on the weighted distance matrix created above. The MEMs are then tested for spatial correlation (Moran's *I*).

```
# Computation of MEM variables (from an object of class listw)
# ------------------------------------------------------------

mite.invdist.MEM <- scores.listw(mite.invdist.lw, echo=TRUE)
summary(mite.invdist.MEM)
mite.invdist.MEM$values
barplot(mite.invdist.MEM$values)

# Test of Moran's I of each eigenvector
(mite.MEM.Moran <- test.scores(mite.invdist.MEM,
mite.invdist.lw, 999))
# MEM with significant spatial correlation
which(mite.MEM.Moran[,2] <= 0.05)
length(which(mite.MEM.Moran[,2] <= 0.05))

  # 31 to 33 MEM variables have significant Moran's I
  # coefficients; among these only MEM 1, 2, 3, 4, 5, 6 and 7
  # (significance of 6 and 7 depending on the permutation result)
  # have positive spatial correlation.
```

```
# Store the MEM vectors in new objects
# All MEM
mite.invdist.MEM.vec <- mite.invdist.MEM$vectors
# MEM with positive spatial correlation
MEM.Moran.pos <- which(mite.MEM.Moran[,1] > -
 1/(nrow(mite.invdist.MEM$vectors)-1))
mite.invdist.MEM.pos <- mite.invdist.MEM.vec[,MEM.Moran.pos]
# MEM with positive *and significant* spatial correlation
MEM.Moran.pos.sig <-
 MEM.Moran.pos[which(mite.MEM.Moran[MEM.Moran.pos,2] <= 0.05)]
mite.invdist.MEM.pos.sig <-
 mite.invdist.MEM.vec[,MEM.Moran.pos.sig]
```

To show that the MEM variables are directly related to Moran's *I*, let us draw a scatterplot of the MEM eigenvalues and their corresponding Moran's *I*:

```
# Plot of MEM eigenvalues vs Moran's I
plot(mite.invdist.MEM$values, mite.MEM.Moran$stat,
 ylab="Moran's I", xlab="Eigenvalues")
text(-1, 0.5, paste("Correlation=", cor(mite.MEM.Moran$stat,
 mite.invdist.MEM$values)))
```

As in the case of the automatic model selection presented before, these MEM variables can now be used as explanatory variables in RDA or multiple regression, in the same way as PCNM variables were.

These are but several examples. We suggest that you explore the manual of the package **spacemakeR**, which presents in great detail the use of many options to construct, present and use various types of connectivity matrices.

7.4.3.4 Controlling for Spatial Correlation Using MEM

Peres-Neto and Legendre (2010) explored the potential use of polynomials and MEM eigenfunctions to control for spatial correlation in statistical tests. Their main conclusion is that MEM, but not polynomials, can adequately achieve this goal. They propose the following procedure: (1) Test for the presence of a spatial structure using all positive MEM variables. (2) If the global test is significant, proceed to forward-select MEM variables, but (a novelty) do this individually for each species, and retain the union of the MEMs selected, i.e. retain all MEMs that have been selected at least once. (3) Proceed to test the species–environment relationships, controlling for spatial correlation by placing the retained MEM variables in a matrix of covariables. The authors demonstrate that this procedure yields correct type I error for tests of significance in linear models, in the presence of spatial correlation.

7.4.3.5 MEM on Sampling Designs with Nested Spatial Scales

The hierarchical structure of many natural entities (e.g. metapopulations or metacommunities; landscapes at various scales) sometimes calls for nested sampling designs. An example is found in Declerck et al. (2011), where the authors studied cladoceran metacommunities in wetland pools found in several valleys of the High Andes. The authors analysed the metacommunity spatial structure among and within valleys by means of a two-level spatial model. The among-valley component was modelled by a set of dummy variables. For the within-valley component, where several pools had been sampled in each valley, a set of MEM variables was computed for each valley. All dummy and MEM variables were assembled into a single staggered matrix. The MEM variables were arranged in blocks corresponding to each valley. Within each block, all pools belonging to other valleys received the value 0, in a way similar to the one presented in Appendix C of Legendre et al. (2010) in the context of space–time analysis. Declerck et al. (2011) provide a function called **`create.MEM.model()`** to construct the staggered spatial matrix from a set of Cartesian coordinates and information about the number of groups and number of sites per group. That function is also available in the electronic material accompanying this book (see Chap. 1).

7.4.4 MEM with Positive or Negative Spatial Correlation: Which Ones Should Be Used?

In the course of the examples above, PCNM and MEM eigenfunctions have been produced, some with positive and some with negative spatial correlation. The question therefore arises: should one use all the (significant) eigenfunctions as explanatory variables in the following regression or canonical analyses, or only those that model positive spatial correlation?

There is no single answer to this question. Ecologically speaking, one is generally more interested in features that are positively correlated at various ranges, simply because they are the signature of contagious processes that are frequent in nature. On the other hand, our experience shows that with real data the significant and negatively correlated variables are either related to very local, almost "accidental" data structures, or they belong to the pure spatial fraction of variation in partitioning, i.e. they correspond to biotic interactions. If these are of interest, then all eigenfunctions should be considered in the analyses.

The original PCNM procedure generates a maximum of $2n/3$ eigenfunctions (n = number of sites), with roughly the first $n/2$ modelling positive spatial correlation, so a forward selection procedure including all variables can be conducted with the Blanchet et al. (2008a) double stopping criterion, which involves the computation

of the R^2_{adj} of the global analysis. In the generalized MEM framework, this is not possible because this method produces $n-1$ spatial variables, which saturate the regression model if they are all considered together. This is why Blanchet et al. proposed to run separate selections on the MEM with positive and negative eigenvalues (usually, the first and the second half of the eigenfunctions), and then apply the Šidák (1967) correction to the probability values: $P_S = 1-(1-P)^k$ where P is the p value to be corrected and k is the number of tests (here $k=2$).

7.4.5 Asymmetric Eigenvector Maps: When Directionality Matters

7.4.5.1 Introduction

The PCNM and MEM analyses presented above are designed for situations where the physical processes generating the response structures (e.g. in communities) do not present any directionality. In other words, the influence of any given point on its surroundings does not depend on the direction.

There are other situations, however, where directionality matters. The most obvious one is the cases of streams or rivers. Consider community effects driven by current: the physical process is geographically asymmetrical, the influence of a site onto another following an upstream–downstream direction. Colonization of the stream network by fish from the river mouth represents a different process, which follows the opposite direction. PCNM or MEM variables are computed on distance or connectivity matrices, where no directionality is specified. Therefore, information about directionality is lost and the modelling, although adequate to reveal major spatial structures, does not exploit all the potential of directional data. Trends do not have to be extracted from the data prior to asymmetric eigenvector maps (AEM) analysis because directional processes are expected to produce trends in the response data; so, a trend is a part of the response data that one wants to model in AEM analysis.

This is the reason why Blanchet et al. (2008b) developed the AEM modelling method. AEM is an eigenfunction-based technique that uses information about the direction of the physical process, plus the same information as MEM (spatial coordinates of sites, connexion diagram, optional weights) if needed. It works best on tree-like structures like river networks or on two-dimensional sampling designs like series of cross-river traps or sampling sites located in a large river or marine current. Depending on the process under study, the origin(s) or root(s) in a river network may be located upstream (e.g. flow of dissolved chemical substances, plankton dispersal) or downstream (fish invasion routes).

For spatial transects or time series, AEM and MEM regression and canonical models are very similar, and in most cases they explain the response data with similar (although not strictly equal) R^2. The AEM eigenfunctions are cosine-like, just like MEM eigenfunctions, although the AEMs have longer wavelengths than MEMs along transects. If the n observations are regularly spaced along the transect and the sampling interval is s, the wavelength λ_i of the AEM with rank i is $\lambda_i = 2ns/i$. AEM analysis should be preferred when modelling gradients and other spatial structures generated by directional physical processes.

AEM analysis was devised for cases where *physical* forces drive the communities in such a way that the causal relationships are directional. This is not the same as a simple ecological gradient, where an ecological factor is spatially structured but the communities can still interact in any direction. In the latter case, PCNM and MEM modelling are appropriate.

7.4.5.2 Principle of the Method

The basic piece of information needed is a table, where each site is described by the connexions (hereafter called "edges", following graph-theory vocabulary) that it has with other sites located in the direction of the root(s) or origin(s) of the directional structure. The result is a rectangular sites-by-edges table **E**, where the sequence of edges connecting each site to the "root" of the network receive code "1" and the others get code "0".

Legendre and Legendre (1998, Section 1.5.7) give an example for fish dispersal from the river mouth in a group of lakes interconnected by a river arborescence. In other cases, for instance a two-dimensional grid consisting of rows of sampling devices placed across a large river or a marine current at regular or irregular intervals, each sampling point may influence (and hence be connected to) the one directly downstream of it, plus the two adjacent to the latter. If the process is assumed to originate upstream, an imaginary point "0" is created upstream of the sampling area, representing the root of the process, with connexions to each of the points in the first row of sites. All edges present in the network are numbered. In table **E**, the rows (i) are the sites and the columns (j) are the edges. The construction rule for AEM is that $E(i,j) = 1$ for all edges j connecting site i to the root (or site 0) of the graph; otherwise, $E(i,j) = 0$.

The edges (columns) of table **E** may be weighted if deemed necessary, e.g. if the transmission of the directional effects are supposed to be easier through some paths than others.

The next step consists in transforming table **E** into eigenfunctions. This can be done in different ways, but the simplest is to compute a PCA of table **E** and use

the matrix of principal components as explanatory variables. The AEM method produces $n-1$ eigenvectors with positive eigenvalues and none with negative eigenvalues. The corresponding eigenfunctions, however, are also divided in two groups depicting positive or negative spatial correlation so that the selection of significant variables must also be run separately for these two groups, in the same way as for MEM variables.

A more detailed explanation about AEM construction is provided by Blanchet et al. (2008b). The authors address the various issues related to edge definition and weighting, which can greatly influence the results of AEM analysis. Blanchet et al. (2011) present three applications to real ecological data.

As a first example, let us construct a fictitious set of AEM variables based on the river arborescence shown by Legendre and Legendre (1998, Section 1.5.7). This example shows how to construct AEM variables in the simplest case, when one can easily produce a matrix of edges by hand. The construction is done by function **aem()** of the package **AEM**.

```
# AEM analysis
# ************

# Coding of a river arborescence.
# See Legendre and Legendre (1998, p. 47).
lake1 <- c(1,0,1,1,0,0,0,0)
lake2 <- c(1,0,1,0,0,0,0,0)
lake3 <- c(1,1,0,0,0,0,0,0)
lake4 <- c(0,0,0,0,1,0,1,1)
lake5 <- c(0,0,0,0,0,1,1,1)
lake6 <- c(0,0,0,0,0,0,0,1)
arbor <- rbind(lake1, lake2, lake3, lake4, lake5, lake6)

# AEM construction
(arbor.aem <- aem(binary.mat=arbor))
arbor.aem.vec <- arbor.aem$vectors

# AEM eigenfunctions can also be obtained directly by singular
# value decomposition (function svd()), which is what the
# function aem() does:
arbor.c = scale(arbor, center=TRUE, scale=FALSE)
arbor.svd = svd(arbor.c)
# Singular values of the previous construction
arbor.svd$d[1:5]
# AEM eigenfunctions of the previous construction
arbor.svd$u[,1:5]
```

Let us now construct AEM variables in a case where the number of data points and edges is too large to allow the use of the simple procedure presented above. The sampling design consists of ten cross-river transects with four traps per transect, and the edges are weighted proportional to inverse squared distance (Fig. 7.10). The procedure involves function **`cell2nb()`** of the package **`spdep`** to construct a list of neighbours from a grid of predefined dimensions.

```
# Coding of sampling design: 10 cross-river transects, 4 traps
# per transect. Edges weighted proportional to inverse squared
# distance.

# X-Y coordinates
xy <- cbind(1:40, expand.grid(1:4, 1:10))

# Object of class nb (spdep) containing links of chess type
# "queen"
nb <- cell2nb(4, 10, "queen")

# Site-by-edges matrix (produces a fictitious object "0")
edge.mat <- build.binary(nb, xy)

# Matrix of Euclidean distances
D1.mat <- as.matrix(dist(xy))

# Extract the edges, remove the ones directly linked to site 0
edges.b <- edge.mat$edges[-1:-4,]

# Construct a vector giving the length of each edge
length.edge <- vector(length=nrow(edges.b))
for(i in 1:nrow(edges.b)){
 length.edge[i] <- D1.mat[edges.b[i,1], edges.b[i,2]]
}

# Weighting of edges based on inverse squared distance
weight.vec <- 1-(length.edge/max(length.edge))^2

# Construction of AEM eigenfunctions from edge.mat,
# of class build.binary
example.AEM <- aem(build.binary=edge.mat, weight=weight.vec,
 rm.link0=TRUE)
example.AEM$values
ex.AEM.vec <- example.AEM$vectors
```

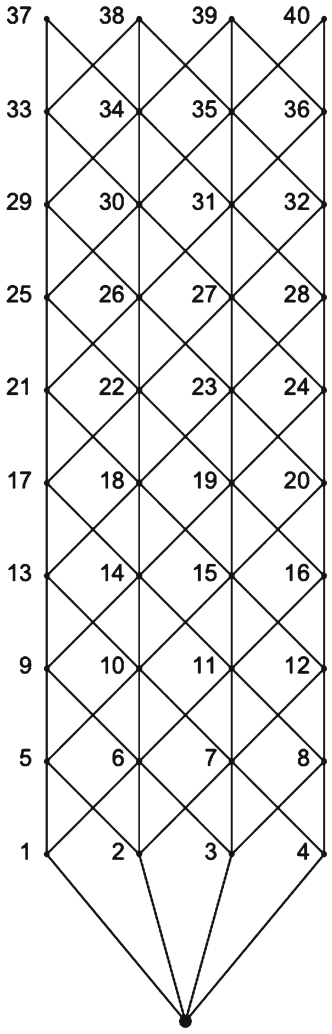

Fig. 7.10 Fictitious directional sampling design for AEM analysis: ten rows of four capture devices along a stream. A site "0" has been added upstream (*bottom* of the figure) to set the direction of the flow

Let us now construct a set of five fictitious species observed at these 40 sites:

```
# Construction of 5 fictitious species

# Two randomly distributed species
sp12 <- matrix(trunc(rnorm(80, 5, 2), 0), 40)
```

```
# One species restricted to the upper half of the stream
sp3 <- c(trunc(rnorm(20, 8, 2.5), 0), rep(0, 20))

# One species restricted to the left-hand half of the transect
sp4 <- t(matrix(c(trunc(rnorm(20, 8, 3), 0), rep(0, 20)), 10))
sp4 <- c(sp4[,1], sp4[,2], sp4[,3], sp4[,4], sp4[,5], sp4[,6],
sp4[,7], sp4[,8], sp4[,9], sp4[,10])

# One species restricted to the 4 upper left-hand sites
sp5 <- c(4, 7, 0, 0, 3, 8, rep(0, 34))

sp <- cbind(sp12, sp3, sp4, sp5)
```

We are ready to proceed with the AEM analysis, using the first half (20, with positive spatial correlation) of the AEM variables generated earlier. Note that four out of five species have a random component, so the actual result of the following AEM analysis will vary from run to run.

```
# Global AEM analysis with 20 first AEM variables (for
# computation of R2a)
# ************************************************
AEM.20 <- rda(sp ~ ., as.data.frame(ex.AEM.vec[,1:20]))
R2a.AEM <- RsquareAdj(AEM.20)$adj.r.squared

AEM.fwd <- forward.sel(sp,ex.AEM.vec, adjR2thresh=R2a.AEM)
(AEM.sign <- sort(AEM.fwd[,2]))
# Write significant AEM in a new object
AEM.sign.vec <- ex.AEM.vec[,c(AEM.sign)]
(sp.AEMsign.rda <- rda(sp ~ .,
 data=as.data.frame(AEM.sign.vec)))  # RDA with signif. AEM
anova.cca(sp.AEMsign.rda)
AEM.rda.axes.test <- anova.cca(sp.AEMsign.rda, by="axis")
# Number of significant axes
(nb.ax.AEM <- length(which(AEM.rda.axes.test[,5] <= 0.05)))

# Plot of the significant canonical axes
AEM.rda.axes <- scores.cca(sp.AEMsign.rda, choices=c(1,2),
 display="lc", scaling=1)
par(mfrow=c(1, nb.ax.AEM))
for(i in 1:nb.ax.AEM) s.value(xy[,c(2,3)], AEM.rda.axes[,i])
```

In most of the runs, this small example shows that AEM analysis reveals the patterns formed by the species present only in the upper half of the stream, as well as the left–right contrast created by the species present only in the left-hand part of

the stream. The pattern of the more restricted species # 5 is less obvious. A PCNM analysis (not shown here) reveals less of the structure and has a lower R^2_{adj}. This stresses the importance of modelling directional processes adequately.

7.5 Another Way to Look at Spatial Structures: Multiscale Ordination

7.5.1 Principle

Wagner (2003, 2004) took an entirely different path towards integration of spatial information into canonical ordination. Under the well-known argument that autocorrelated residuals can alter the results of statistical tests, she introduced geostatistical methods to devise diagnostic tools allowing the partitioning of ordination results into distance classes, the distinction between induced spatial dependence and spatial autocorrelation, and the use of variograms to check important assumptions, such as independence of residuals and stationarity. The principle of multiscale ordination (MSO) is the following[3]:

- Analyse the species by RDA. The explanatory variables can be of any kind (environmental, spatial, …). This provides the matrix of fitted values and its eigenvectors, as well as the matrix of residuals and its eigenvectors.
- By way of a variogram matrix computed for the fitted values, obtain the spatial variance profiles of the canonical ordination axes (see below).
- By way of a variogram matrix computed for the residuals, obtain the spatial variance profiles of the residual ordination axes.
- Plot the variograms of the explained and residual variances. Permutation tests may be used to identify significant spatial correlation in the distance classes.

A variogram matrix is a three-dimensional array containing a separate variance–covariance matrix for each distance class (Wagner, 2003, Fig. 2). The diagonal of each matrix quantifies the contribution of the corresponding distance class to the variance of the data. MSO computes a variogram matrix on the fitted values of a constrained ordination, thereby allowing its spatial decomposition. Multiplying this variogram matrix with the matrix of constrained eigenvectors provides the spatial decomposition of each eigenvalue (variance profiles). The same holds for the residuals.

[3] Wagner (2004) describes the method for CCA, but the principle is the same for RDA.

7.5.2 Application to the Mite Data: Exploratory Approach

Let us use the oribatid mite data as an example. Wagner (2004) also used these data, but in a CCA context so that the results will differ. MSO can be computed using function **mso()** of the package **vegan**. This function uses a result object produced by functions **cca()** or **rda()**, plus the table of geographical coordinates and a value for the interval size (argument "grain") of the distance classes of the variograms. The first example applies MCO in the exploratory way proposed by Wagner. An MSO plot of direct ordination can show whether the spatial structure in the response data can be explained by the explanatory (environmental) variables alone. In such a case, no detrending is necessary (H. Wagner, pers. comm.), but the confidence interval of the variogram is indicative only, since a variogram should be computed on stationary data.

Hereunder, MSO is run using the RDA result of the Hellinger-transformed oribatid mite data explained by the environmental variables. The "grain" of the variogram (size of a distance class) is chosen to be the truncation threshold used in the PCNM analysis, 1.011187.

```
# Multiscale ordination (MSO)
# ***************************

# MSO of the undetrended mite data vs environment RDA
# ---------------------------------------------------
mite.undet.env.rda <- rda(mite.h, mite.env2)

mite.env.rda.mso <- mso(mite.undet.env.rda, mite.xy, grain=dmin,
  perm=999)
msoplot(mite.env.rda.mso, alpha=0.05/7)
mite.env.rda.mso
```

The resulting plot (Fig. 7.11) provides several informations. In the upper part of the diagram, the dashed line with the crosses represents the sum of the explained and residual empirical variograms. The continuous lines represent the confidence envelopes of the variogram of the data matrix. The monotonic increase of the dashed line is the signature of the strong linear gradient present in the data. Note, however, that the variogram of the residuals (bottom of the graph) shows no distance class with significant spatial correlation (after a global Bonferroni correction for seven simultaneous tests: rejection threshold divided by the number of classes), and that variogram is essentially flat. This means that the broad-scale linear gradient is well explained by the environmental variables.

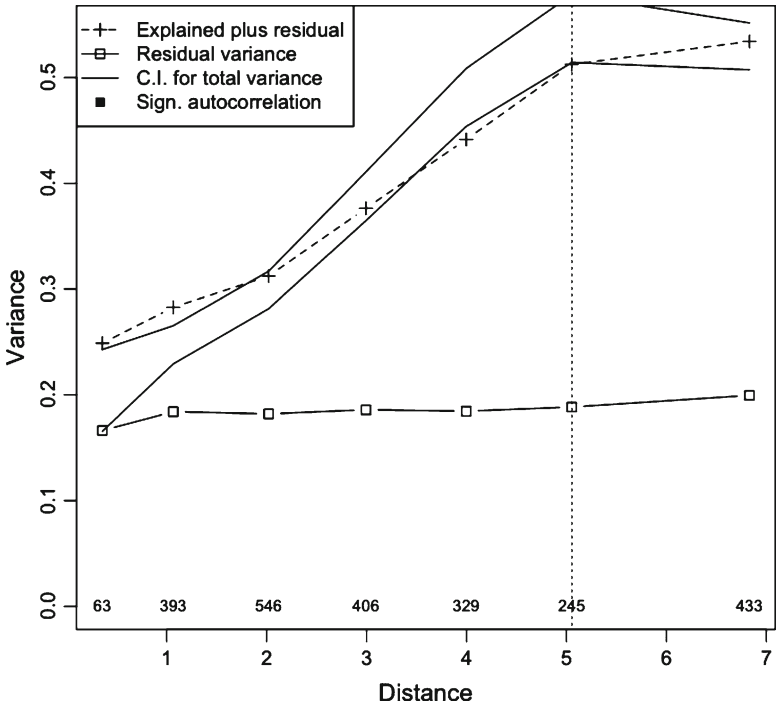

Fig. 7.11 Plot of the MSO of a RDA of the Hellinger-transformed oribatid mite data explained by the environmental variables. Explanations: see text

However, an intriguing feature appears. When the species–environment correlations do not vary with scale, the dashed line remains within the boundaries of the confidence envelopes. This is not the case here (see classes 1, 2 and 5), suggesting that it is not appropriate to run a non-spatial, global species–environment analysis with the implicit assumption that the relationships are scale-invariant. On the contrary, we can expect the regression parameters to vary with scale, so that a global estimation is meaningless unless one controls for the regional scale spatial structure causing the problem.

As an attempt in this direction, let us run an MSO on a partial RDA of the mite species explained by the environment, controlling for the spatial structure, here represented by the seven MEM variables of our best model.

```
# MSO of the undetrended mite data vs environment RDA,
# controlling for MEM
# ----------------------------------------------------
mite.undet.env.MEM <- rda(mite.h, mite.env2,
as.data.frame(MEM.select))
mite.env.MEM.mso <- mso(mite.undet.env.MEM, mite.xy, grain=dmin,
  perm=999)
msoplot(mite.env.MEM.mso, alpha=0.05/7)
mite.env.MEM.mso
```

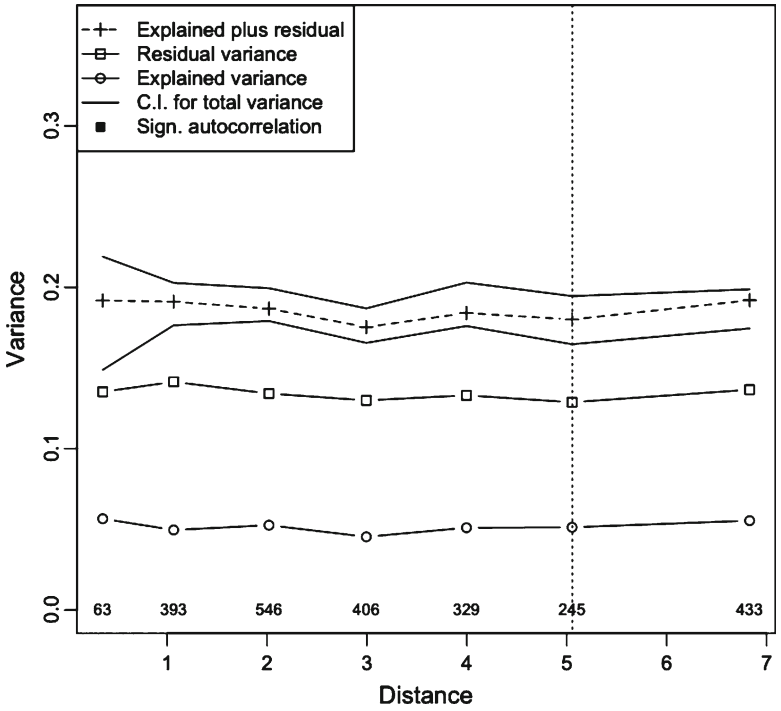

Fig. 7.12 Plot of the MSO of a RDA of the Hellinger-transformed oribatid mite data explained by the environmental variables, controlling for spatial structure (seven MEM variables)

Figure 7.12 shows that the problem of scale-dependence in the model has been properly addressed. There is no spatial correlation in the residuals, and the variogram of the residual species–environment relationship (after taking the MEM spatial structure into account) stays within the confidence interval across all scales. Furthermore, the MEM variables have also removed the major gradient from the data, resulting in a globally flat empirical variogram. The console message stating that the "Error variance of regression model [is] underestimated by -Inf percent"

actually refers to the difference between the total residual variance and the sill of the residual variance. When the value is negative (and extreme in this case), the absence of significant autocorrelation causes an underestimation of the global error value of the regressions. A positive value (e.g. 10%), which could occur if the residuals were significantly autocorrelated, would act as a warning that the condition of independent residuals is violated, thereby invalidating the statistical tests (see Sect. 7.2.2).

7.5.3 *Application to the Detrended Mite and Environmental Data*

Let us apply an MSO analysis on detrended data, as an effort to meet the conditions of application of the calculation of the variogram confidence intervals. We know from Sect. 7.4.2.5 that there is a significant spatial structure only in the *Y* direction. We shall therefore detrend the mite and environmental data on the *Y* coordinate before running the RDA.

```
# MSO on detrended mite and environmental data
# -------------------------------------------

# Detrend mite data on Y coordinate
mite.h.det2 <- resid(lm(as.matrix(mite.h) ~ mite.xy[,2]))

# Detrend environmental data on Y coordinate
env2.det <- resid(lm(as.matrix(mite.env2) ~ mite.xy[,2]))

# RDA and MSO
mitedet.envdet.rda <- rda(mite.h.det2, env2.det)
miteenvdet.rda.mso <- mso(mitedet.envdet.rda, mite.xy,
  grain=dmin, perm=999)
msoplot(miteenvdet.rda.mso, alpha=0.05/7)
miteenvdet.rda.mso
```

The result (Fig. 7.13) tells us a similar story, less the broad-scale gradient which has been removed prior to the analysis by detrending. The residual variance shows no spatial correlation, and the second, fourth and fifth class of the variogram of explained plus residual data fall outside the confidence interval. So the overall variogram shows no trend, but some regional spatial variance is present. Can the MEM control successfully for this spatial variance?

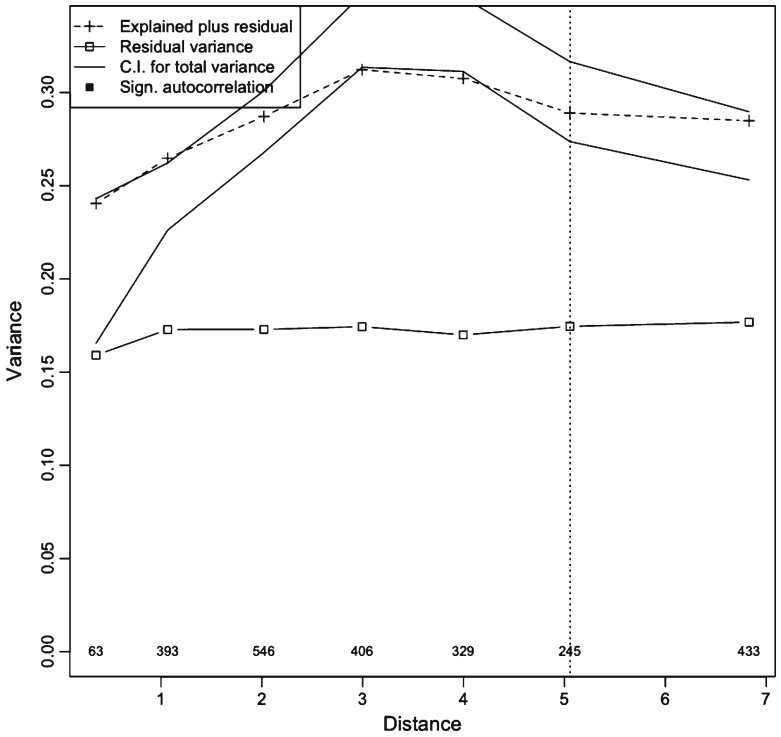

Fig. 7.13 Plot of the MSO of a RDA of the Hellinger-transformed and detrended oribatid mite data explained by the detrended environmental variables. Explanations: see text

```
# MSO of the detrended mite data vs environment RDA, controlling
# for MEM
# ------------------------------------------------------------------

mite.det.env.MEM <- rda(mite.h.det2, env2.det,
as.data.frame(MEM.select))
mite.env.MEM.mso <- mso(mite.det.env.MEM, mite.xy, grain=dmin,
  perm=999)
msoplot(mite.env.MEM.mso, alpha=0.05/7)
mite.env.MEM.mso
```

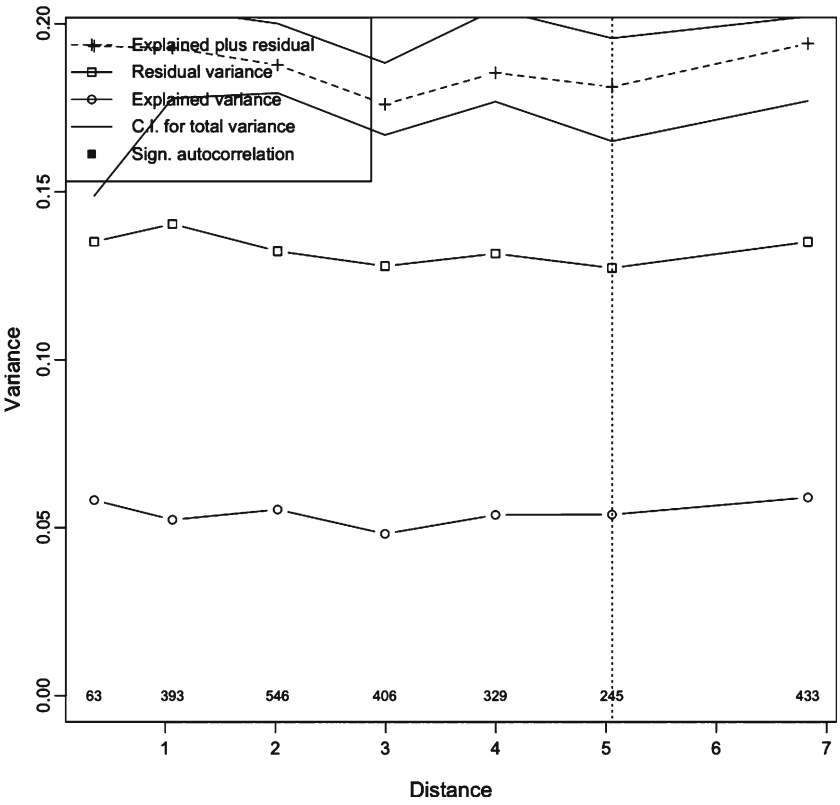

Fig. 7.14 Plot of the MSO of a RDA of the Hellinger-transformed and detrended oribatid mite data explained by the detrended environmental variables, controlling for spatial structure (seven MEM variables). Further explanations: see text

The answer is "yes" (Fig. 7.14). As in the undetrended example, one can see no spatial variance in the residuals or in the data. Compare with Fig. 7.12: the variograms are very similar (although the default graphical output provides a different ordinate scale). The MEM variables have successfully controlled for the spatial variance unexplained by the environmental data.

This example shows the potential of combining multivariate geostatistical methods with canonical ordination when the aim of the study is to test for and model species–environment relationships while discriminating between the two major sources of concern related to spatial structures: spatial dependence (7.1) and spatial autocorrelation (7.2). Some aspects of this approach remain to be explored, however. Wagner (2004) notes "*an important discrepancy between the results presented*

here and those by Borcard et al. (1992). Borcard found that 12.2% of the total inertia was spatially structured but could not be explained by the environmental variables. In the spatial partitioning of CCA results by multi-scale ordination (MSO), however, spatial autocorrelation appeared to be limited to distances smaller than 0.75 m, and there was no evidence of any cyclic pattern that could account for such a large portion of inertia. The large portion of nonenvironmental spatial structure identified by Borcard et al. (1992) may partly be due to a confounding of the effects of space and environment (Méot et al. 1998)". Arguing from an opposite point of view, we believe that the pure spatial structures revealed by canonical ordination (and especially in the PCNM and MEM framework which would give an even larger pure spatial fraction) are real and not due to confounding effects. In the latter case, they would have shown up in the common fraction of variation, not the pure spatial fraction. The question is rather: why did the MSO *not* reveal these structures? This may be due to the fact that no formal way of quantifying variance components in MSO has been devised as yet (H. Wagner, pers. comm.). However, this does by no means invalidate the MSO approach which, combined with the powerful tools developed in this chapter, increases our control over the complex process of extracting meaningful spatial information from ecological data.

7.6 Conclusion

Spatial analysis of ecological data has undergone huge developments during the last three decades. The paradigm shift announced by Legendre (1993) has been accompanied by an increasing awareness, not only of the importance of spatial structures per se, but also of the need for refined modelling tools to identify, represent and explain the complex structures by which ecological interactions manifest themselves in living communities. While an entire family of techniques aimed at prediction and mapping has been developed in the field of geostatistics and some of them can be applied to ecological problems, the specific questions and data in ecology demanded other approaches more directly related to the multivariate structure of communities and their relationship to the environment. We have presented the most important among them in this chapter, encouraging readers to apply them to their own data in a creative way. The multiscale nature of ecological problems can now be addressed in a much deeper way than before, and the authors of the methods are themselves constantly surprised at the range of applications ecologists make of their statistical offsprings. Many more developments will certainly be made in the forthcoming years, and we wish to conclude by inviting the readers to participate in this effort, both by asking new and challenging ecological questions and by devoting themselves to the exciting task of methodological development.