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**), with its visualization tools and simple statistics, 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 will explore the classical Doubs River 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 data used here are available in a .RData file found among the files provided with the book; see Chap. 1.

D. Borcard et al., *Numerical Ecology with R*, Use R!,
https://doi.org/10.1007/978-3-319-71404-2_2

```
# Load required packages
library(vegan)
library(RgoogleMaps)
library(googleVis)
library(labdsv)

# Source additional functions that will be used later in this
# Chapter. Our scripts assume that files to be read are in
# the working directory.
source("panelutils.R")

# Load the data. File Doubs.Rdata is assumed to be
# in the working directory
load("Doubs.RData")

# The file Doubs.RData contains the following objects:
#    spe: species (community) data frame (fish abundances)
#    env: environmental data frame
#    spa: spatial data frame - cartesian coordinates
#    fishtraits: functional traits of fish species
#    latlong: spatial data frame - latitude and longitude
```

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()`**.

Although it is not necessary, we strongly recommend that you use RStudio as script manager, which adds many interesting features to standard text editors. The R code in the companion materials of this book is optimized for RStudio and complies with the R Core Team's guidelines for good practices in R programming. Once all necessary files are placed in the same folder and RStudio is configured to run R scripts, just double-click on an R script file and the corresponding folder will be automatically defined as the current working directory.

Users of the standard ***R*** *console can use the* ***R*** *built-in text editor to write* ***R*** *code and run any selected portion using easy keyboard commands (<Control+Return> or <Command+Return> depending on the machine you are using). To open a new file, click on the File menu, then click on New script. Dragging an* ***R*** *script, for example our file "chap2.R", onto the R icon, will automatically open it in a new file managed by the* ***R*** *text editor.*

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 will first focus on the community data (object **`spe`** loaded as an element of the Doubs.RData file above). Verneaux used a semi-quantitative, species-specific, abundance scale (0–5), so that comparisons between

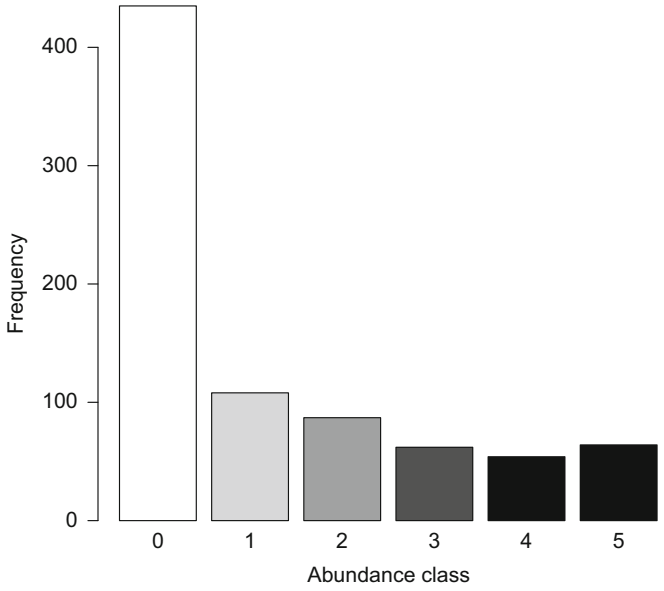

Fig. 2.1 Barplot of abundance classes

species abundances will make sense. The maximum value, 5, corresponds to the class with the maximum number of individuals captured by electrical fishing in the Doubs River and its tributaries (i.e. not only in this data set) by Verneaux. Therefore, species-specific codes cannot be understood as unbiased estimates of the true abundances (number or density of individuals) or biomasses at the sites.

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

```
## Exploration of a data frame using basic R 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 6 lines
tail(spe)                    # Display only the last 6 rows
nrow(spe)                    # Number of rows (sites)
ncol(spe)                    # Number of columns (species)
dim(spe)                     # Dimensions of the data frame (rows,
                             # columns)
```

```r
colnames(spe)              # Column labels (descriptors = species)
rownames(spe)              # Row labels (objects = sites)
summary(spe)               # Descriptive statistics for columns

## Overall distribution of abundances (dominance codes)
# Minimum and maximum of abundance values in the whole data set
range(spe)
# Minimum and maximum value for each species
apply(spe, 2, range)
# Count the cases for each abundance class
(ab <- table(unlist(spe)))
# 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))
```

Hint *Observe how the shades of grey of the bars have been defined in the function* **`barplot()`**. *The argument* `col = gray(5 : 0 / 5)` *means "I want five shades of grey with levels ranging from 5/5 (i.e., white) to 0/5 (black)".*

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

2.2.3 Species Data: A Closer Look

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

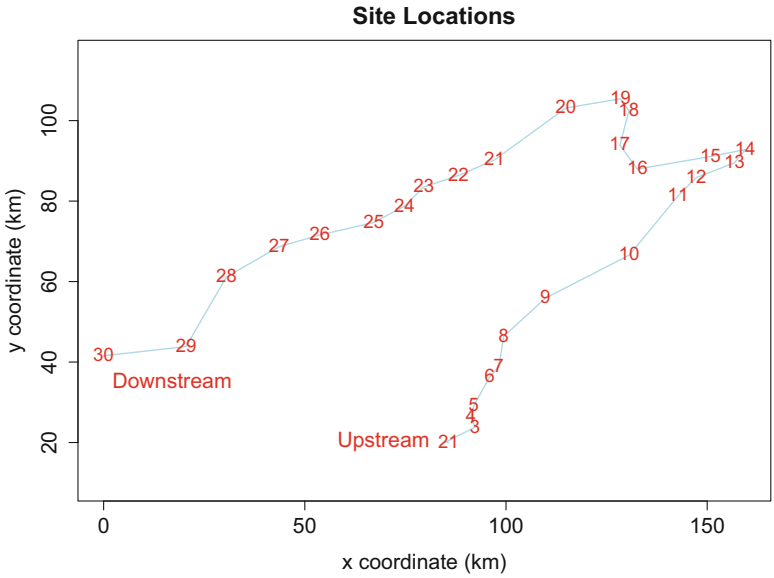

Fig. 2.2 Map of the 30 sampling sites along the Doubs River. Sites 1 and 2 are very close to each other

```
## Map of the locations of the sites
# 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 along the Doubs River
lines(spa, col = "light blue")
# Add the site labels
text(spa, row.names(spa), cex = 0.8, col = "red")
# Add text blocks
text(68, 20, "Upstream", cex = 1.2, col = "red")
text(15, 35, "Downstream", cex = 1.2, col = "red")
```

When the data set covers a sufficiently large area, it is possible to project the sites onto a Google Maps® map:

```
## Sites projected onto a Google Maps® background
# By default the plot method of the googleVis package uses
# the standard browser to display its output.
nom <- latlong$Site
latlong2 <- paste(latlong$LatitudeN, latlong$LongitudeE, sep = ":")
df <- data.frame(latlong2, nom, stringsAsFactors = FALSE)

mymap1 <- gvisMap(df,
  locationvar = "latlong2",
  tipvar = "nom",
  options = list(showTip = TRUE)
)
plot(mymap1)
```

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

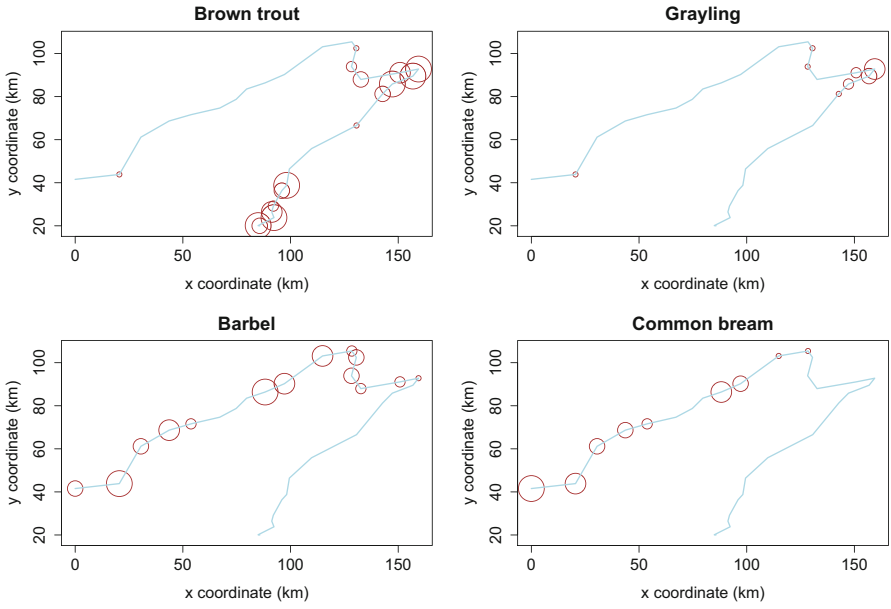

Fig. 2.3 Bubble maps of the abundances of four fish species

```
## Maps of some fish species
# Divide the plot window into 4 frames, 2 per row
par(mfrow = c(2,2))
# Plot four species
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  col = "brown",
  cex = spe$Satr,
  main = "Brown trout",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  col = "brown",
  cex = spe$Thth,
  main = "Grayling",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  col = "brown",
  cex = spe$Baba,
  main = "Barbel",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  col = "brown",
  cex = spe$Abbr,
  main = "Common bream",
  xlab = "x coordinate (km)",
  ylab = "y coordinate (km)"
)
lines(spa, col = "light blue")
```

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 belonging to the* `spe` *data frame, i.e., the abundances of a given species (e.g.,* `cex = spe$Satr`*). 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 list of arguments has been simplified by replacing the first two arguments for horizontal and vertical axes by the name of the spa data frame.*

From these maps you can understand why Verneaux chose these four species as ecological indicators. More about the environmental conditions later.

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

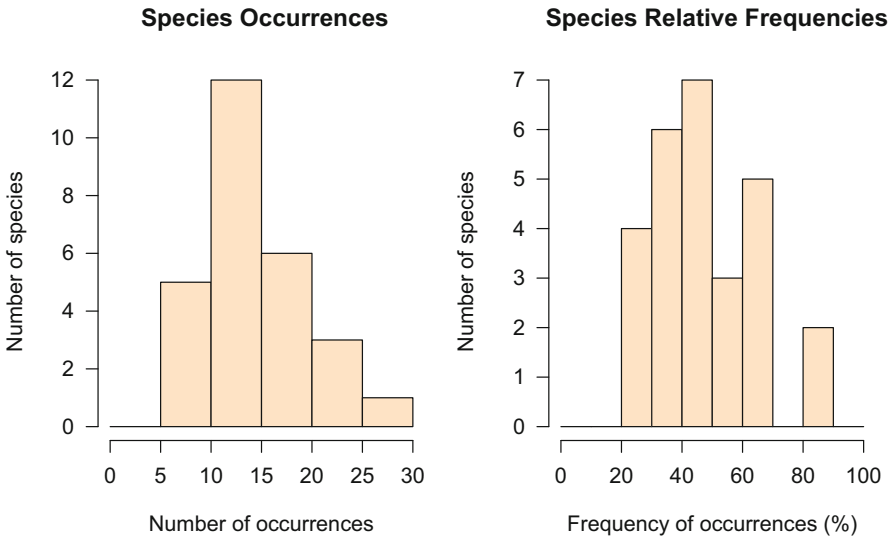

Fig. 2.4 Frequency histograms: species occurrences and relative frequencies at the 30 sites

```
## 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 the sorted output to 1 digit
round(sort(spe.relf), 1)
# Plot the histograms
par(mfrow = c(1,2))
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.*

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):

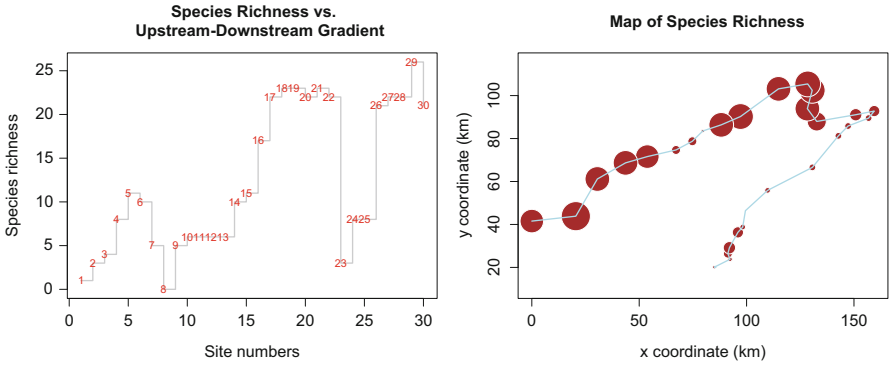

Fig. 2.5 Species richness along the river

```
## 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))
# 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 = "Site numbers",
  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")
```

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

Can you identify the richness hotspots along the river?

More elaborate measures of diversity will be presented in Chap. 8.

2.2.4 Ecological 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. Standardization to z-scores (i.e., centring and reduction) and ranging (to a [0,1] interval) make variables dimensionless. Following that, their variances can be added, e.g. in principal component analysis (see Chap. 5);
- Transform the variables to have a normal (or at least a symmetric) distribution and stabilize their variances (through square root, fourth root, log transformations, etc.);
- Make the relationships among variables linear (e.g., log-transform the response variable if the relationship is exponential);
- Modify the weights of the variables or objects prior to a multivariate analysis, e.g., give the same variance to all variables, or 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 (two or more ordered classes) variables, and restricted to positive or null values (zero meaning absence). For these, **simple transformations** can be used to reduce the importance of observations with very high values; **`sqrt()`** (square root), **`^0.25`** (fourth root), or **`log1p()`** ($\log(y + 1)$ to keep absences as zeros) are commonly applied **R** functions (see also Chap. 3). 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** refers to transformations that have the objective to make the rows or columns of the data table comparable to one another because they will have acquired some property. In contrast to simple transformations such as square root, log or presence-absence, the values are not transformed individually but relative to other values in the data table. Standardizations can be done relative to sites (e.g. relative abundances per site) or

species (abundances scaled with respect to the species maximum abundance or its total abundance), or simultaneously to both site and species totals (with the chi-square transformation), depending on the focus of the analysis.

Note that **`decostand()`** has a `log` argument for logarithmic transformation. But here the transformation is *log_b(y)* + 1 for $y > 0$, where b is the base of the logarithm. Zeros remain untouched. The base of the logarithm is provided by the argument `logbase`. This transformation, proposed by Anderson et al. (2006), is not equivalent to $\log(y + 1)$. Increasing the value of the base increases the severity of the downscaling of large values.

Here are some examples of data standardization illustrated by boxplots (Fig. 2.6).

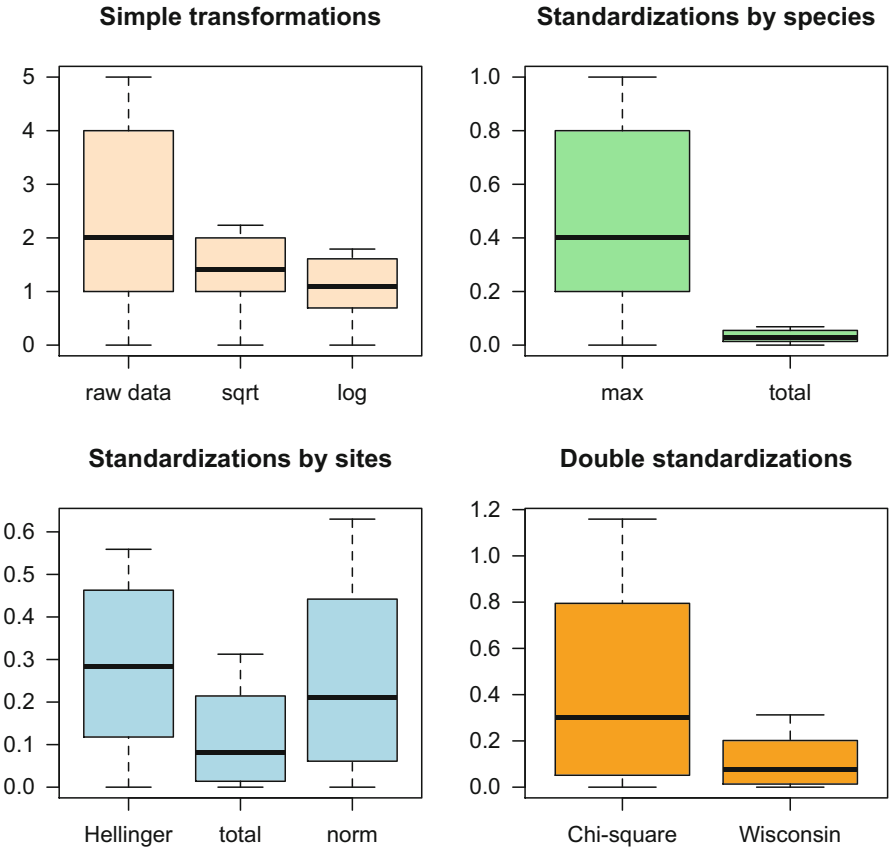

Fig. 2.6 Boxplots of transformed abundances of a common species, *Barbatula barbatula* (stone loach)

```
# 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]

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

> *Did the scaling work properly? It is good to keep an eye on your results by producing a plot or computing summary statistics.*

```
# Scale abundances by dividing them by the species totals
# (relative abundance by species)
# Note: here, override the default MARGIN = 1 argument of "total"
spe.relsp <- decostand(spe, "total", MARGIN = 2)
spe.relsp[1:5, 2:4]
# Display the sum by column
# Classical: apply(spe.relsp, 2, sum)
colSums(spe.relsp)

## Standardization by rows (sites)
# Scale abundances by dividing them by the site totals
# (profiles of relative abundance by site)
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
rowSums(spe.rel)

# Give a length (norm) of 1 to each row vector
# This is called the chord transformation
spe.norm <- decostand(spe, "normalize") # default MARGIN = 1
spe.norm[1:5, 2:4]
```

```
# Verify the norm of the transformed row vectors
# Write a 1-line function that computes the norm of vector x
vec.norm <- function(x) sqrt(sum(x ^ 2))
# Then, apply that function to the rows of matrix spe.norm
apply(spe.norm, 1, vec.norm)
```

The scaling above is called the 'chord transformation': the Euclidean distance function applied to chord-transformed data produces a chord distance matrix (Chap. 3). The chord transformation is useful prior to PCA and RDA (Chap. 5 and 6) and k-means partitioning (Chap. 4). The chord transformation can also be applied to log-transformed data (see Chap. 3).

```
# Compute square root of relative abundances per site
spe.hel <- decostand(spe, "hellinger")
spe.hel[1:5, 2:4]
# Check the norm of row vectors
apply(spe.hel, 1, vec.norm)
```

This is called the Hellinger transformation. The Euclidean distance function applied to Hellinger-transformed data produces a Hellinger distance matrix (Chap. 3). The Hellinger transformation is useful prior to PCA and RDA (Chap. 5 and 6) and k-means partitioning (Chap. 4).

Note: the Hellinger transformation can also be obtained by applying the chord transformation to square-root-transformed species data.

```
## Double standardization by columns and rows
# 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, ]
# Note: decostand produced values of 0 for 0/0 instead of NaN
```

The Euclidean distance function applied to chi-square-transformed data produces a chi-square distance matrix (Chap. 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
# (the stone loach, species #4)
par(mfrow = c(2,2))
boxplot(spe$Babl,
  sqrt(spe$Babl),
  log1p(spe$Babl),
  las = 1,
  main = "Simple transformations",
  names = c("raw data", "sqrt", "log"),
  col = "bisque"
)
boxplot(spe.scal$Babl,
  spe.relsp$Babl,
  las = 1,
  main = "Standardizations by species",
  names = c("max", "total"),
  col = "lightgreen"
)
boxplot(spe.hel$Babl,
  spe.rel$Babl,
  spe.norm$Babl,
  las = 1,
  main = "Standardizations by sites",
  names = c("Hellinger", "total", "norm"),
  col = "lightblue"
)
boxplot(spe.chi$Babl,
  spe.wis$Babl,
  las = 1,
  main = "Double standardizations",
  names = c("Chi-square", "Wisconsin"),
  col = "orange"
)
```

Hint *Take a look at the line:* `vec.norm <- function(x) sqrt(sum(x^2))`. *It is an example of a small function built on the fly to fill a gap in the functions available in 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.*

Compare the effects of these transformations and standardizations on the ranges and distributions of the scaled abundances.

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

```
## Plot raw and transformed abundances along the upstream-
## downstream river gradient
par(mfrow = c(2, 2))
plot(env$dfs,
  spe$Satr,
  type = "l",
  col = 4,
  main = "Raw data",
  xlab = "Distance from the source [km]",
  ylab = "Raw abundance code"
)
lines(env$dfs, spe$Thth, col = 3)
lines(env$dfs, spe$Baba, col = "orange")
lines(env$dfs, spe$Abbr, col = 2)
lines(env$dfs, spe$Babl, col = 1, lty = "dotted")

plot(env$dfs,
  spe.scal$Satr,
  type = "l",
  col = 4,
  main = "Species abundances ranged
  by maximum",
  xlab = "Distance from the source [km]",
  ylab = "Ranged abundance"
)
lines(env$dfs, spe.scal$Thth, col = 3)
lines(env$dfs, spe.scal$Baba, col = "orange")
lines(env$dfs, spe.scal$Abbr, col = 2)
lines(env$dfs, spe.scal$Babl, col = 1, lty = "dotted")

plot(env$dfs,
  spe.hel$Satr,
  type = "l",
  col = 4,
  main =  "Hellinger-transformed abundances",
  xlab = "Distance from the source [km]",
  ylab = "Standardized abundance"
)
lines(env$dfs, spe.hel$Thth, col = 3)
lines(env$dfs, spe.hel$Baba, col = "orange")
lines(env$dfs, spe.hel$Abbr, col = 2)
lines(env$dfs, spe.hel$Babl, col = 1, lty = "dotted")

plot(env$dfs,
```

```
  spe.chi$Satr,
  type = "l",
  col = 4,
  main = "Chi-square-transformed abundances",
  xlab = "Distance from the source [km]",
  ylab = "Standardized abundance"
)
lines(env$dfs, spe.chi$Thth, col = 3)
lines(env$dfs, spe.chi$Baba, col = "orange")
lines(env$dfs, spe.chi$Abbr, col = 2)
lines(env$dfs, spe.chi$Babl, 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 graphs and explain the differences.

In some cases (often vegetation studies), data are collected using abundance scales that are meant to represent specific properties: number of individuals (abundance classes), cover (dominance classes), or both (e.g. Braun-Blanquet abundance-dominance scale). The scales being ordinal and somewhat arbitrary, the resulting data do not easily lend themselves to a simple transformation. In such cases one may have to convert scales by attributing values according to the data at hand. For discrete scales it can be done by function **`vegtrans()`** of package **`labdsv`**.

For example, assuming we knew how to convert the fish abundance codes (ranging from 0 to 5 in our **`spe`** dataset) to average numbers of individuals, we could do it by providing two vectors, one with the current scale and one with the converted scale. **Beware**: this would not make sense for this fish data set, whose abundances are species-specific (see Sect. 2.2).

```
## Conversion of the fish abundance using an arbitrary scale
current <- c(0, 1, 2, 3, 4, 5)
converted <- c(0, 1 ,5, 10, 20, 50)
spe.conv <- vegtrans(spe, current, converted)
```

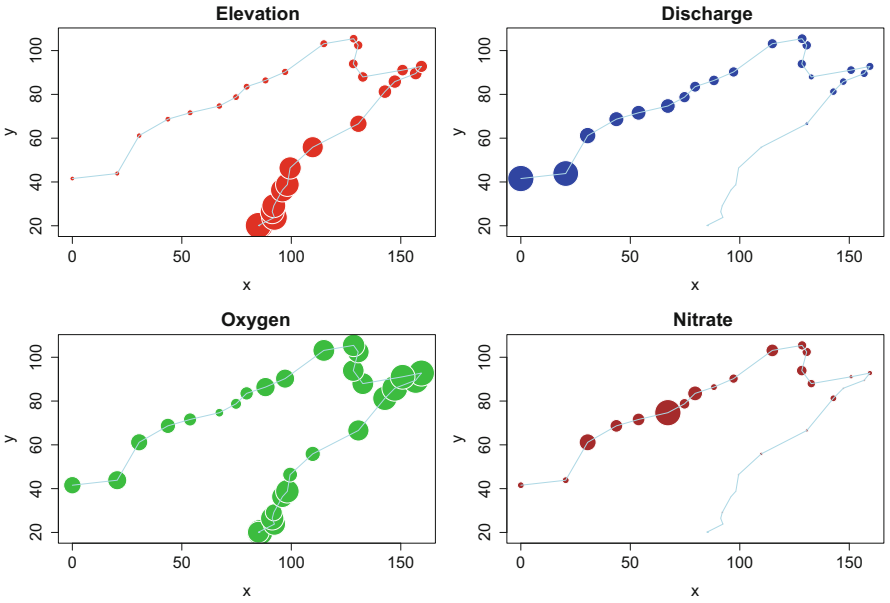

Fig. 2.7 Bubble maps of four environmental variables

2.2.5 *Environmental Data*

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

First, go back to Sect. 2.2 and apply the basic functions presented there to the object **`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):

```
par(mfrow = c(2, 2))
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  main = "Elevation",
  pch = 21,
  col = "white",
  bg = "red",
  cex = 5 * env$ele / max(env$ele),
  xlab = "x",
  ylab = "y"
)
```

```
lines(spa, col = "light blue")
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  main = "Discharge",
  pch = 21,
  col = "white",
  bg = "blue",
  cex = 5 * env$dis / max(env$dis),
  xlab = "x",
  ylab = "y"
)
lines(spa, col = "light blue")
plot(spa,
  asp = 1,
  cex.axis = 0.8,
  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,
  cex.axis = 0.8,
  main = "Nitrate",
  pch = 21,
  col = "white",
  bg = "brown",
  cex = 5 * env$nit / max(env$nit),
  xlab = "x",
  ylab = "y"
)
lines(spa, col = "light blue")
```

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.*

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

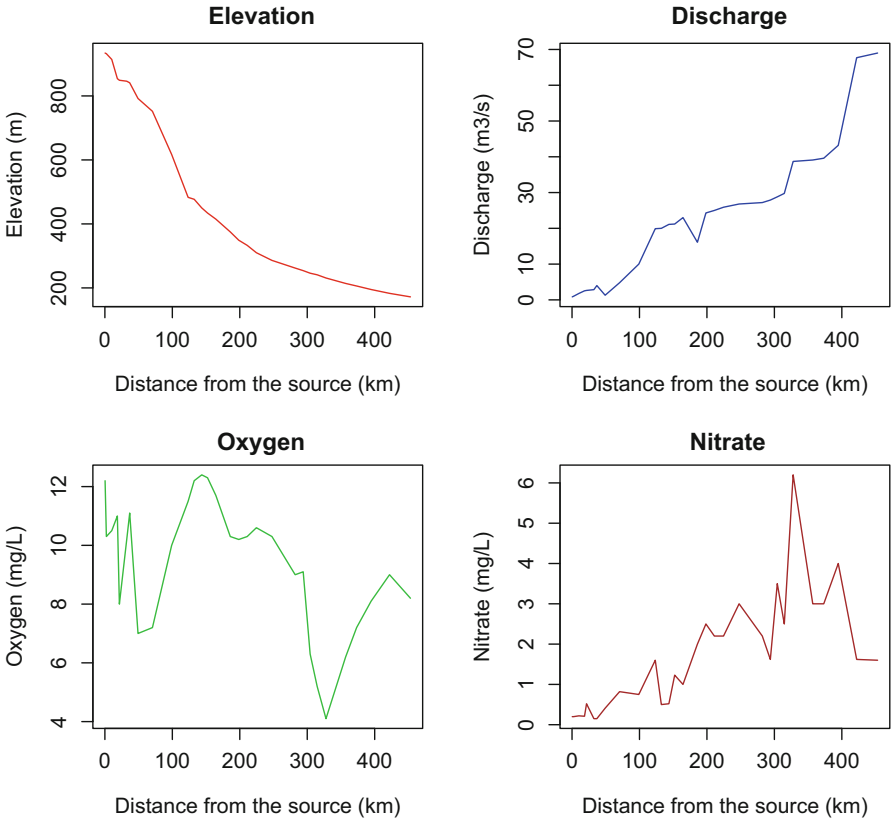

Fig. 2.8 Line plots of four environmental variables

Now, examine the variation of some descriptors along the river by means of line plots (Fig. 2.8):

```
## Line plots
par(mfrow = c(2, 2))
plot(env$dfs, env$ele,
  type = "l",
  xlab = "Distance from the source (km)",
  ylab = "Elevation (m)",
  col = "red", main = "Elevation"
)
```

```
plot(env$dfs, env$dis,
  type = "l",
  xlab = "Distance from the source (km)",
  ylab = "Discharge (m3/s)",
  col = "blue",
  main = "Discharge"
)
plot(env$dfs, env$oxy,
  type = "l",
  xlab = "Distance from the source (km)",
  ylab = "Oxygen (mg/L)",
  col = "green3",
  main = "Oxygen"
)
plot(env$dfs, env$nit,
  type = "l",
  xlab = "Distance from the source (km)",
  ylab = "Nitrate (mg/L)",
  col = "brown",
  main = "Nitrate"
)
```

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 found in the **`panelutils.R`** script.

```
## Scatter plots for all pairs of environmental variables
# Bivariate plots with histograms on the diagonal and smooth
# fitted curves
pairs(env,
  panel = panel.smooth,
  diag.panel = panel.hist,
  main = "Bivariate Plots with Histograms and Smooth Curves"
)
```

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

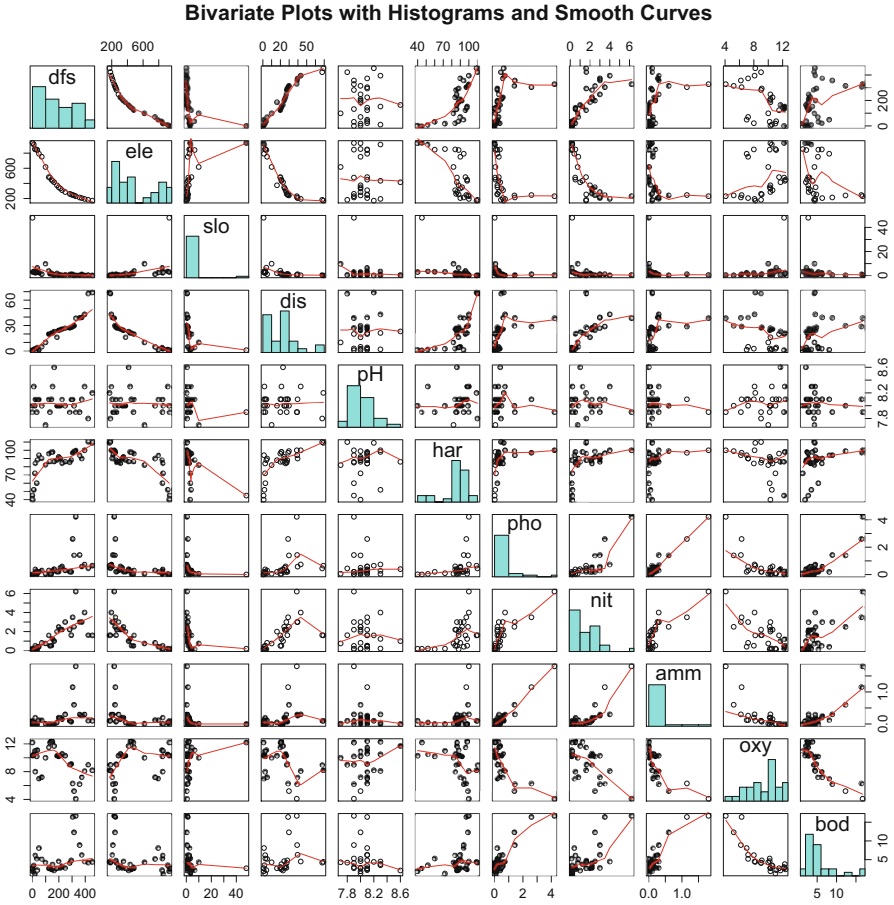

Fig. 2.9 Scatter plots between all pairs of environmental variables with LOWESS smoothers

> *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?*

Simple transformations, such as the log transformation, can be used to improve the distributions of some variables (make them more symmetrical and closer to normal distributions). 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).

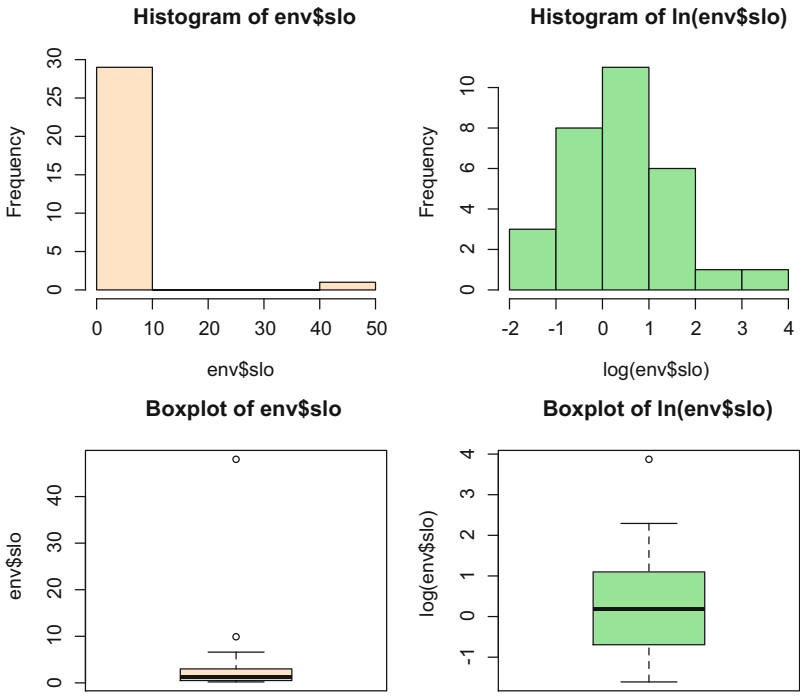

Fig. 2.10 Histograms and boxplots of the untransformed (left) and log-transformed (right) "slope" variable

```
## Simple transformation of an environmental variable
range(env$slo)
# Log-transformation of the slope variable (y = ln(x))
# Compare histograms and boxplots of the raw and transformed values
par(mfrow = c(2, 2))
hist(env$slo,
  col = "bisque",
  right = FALSE
)
hist(log(env$slo),
  col = "light green",
  right = FALSE,
  main = "Histogram of ln(env$slo)"
)
boxplot(env$slo,
  col = "bisque",
  main = "Boxplot of env$slo",
  ylab = "env$slo"
)
boxplot(log(env$slo),
  col = "light green",
  main = "Boxplot of ln(env$slo)",
  ylab = "log(env$slo)"
)
```

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

```
## Standardization of all environmental variables
# Center and scale = standardize the 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))
```

2.3 Conclusion

The tools presented in this chapter allow researchers to get a general impression of their data. Although you will see much more elaborate analyses in the following 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 and will help in the correct selection of more advanced analyses. 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 scientists who are eager to jump to more sophisticated analyses. We hope we have convinced you that it should have an important place in the toolbox of ecologists.