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?
```

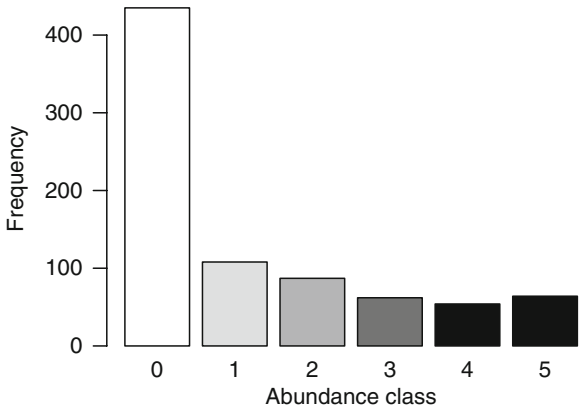

Fig. 2.1 Barplot of abundance classes

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")
```

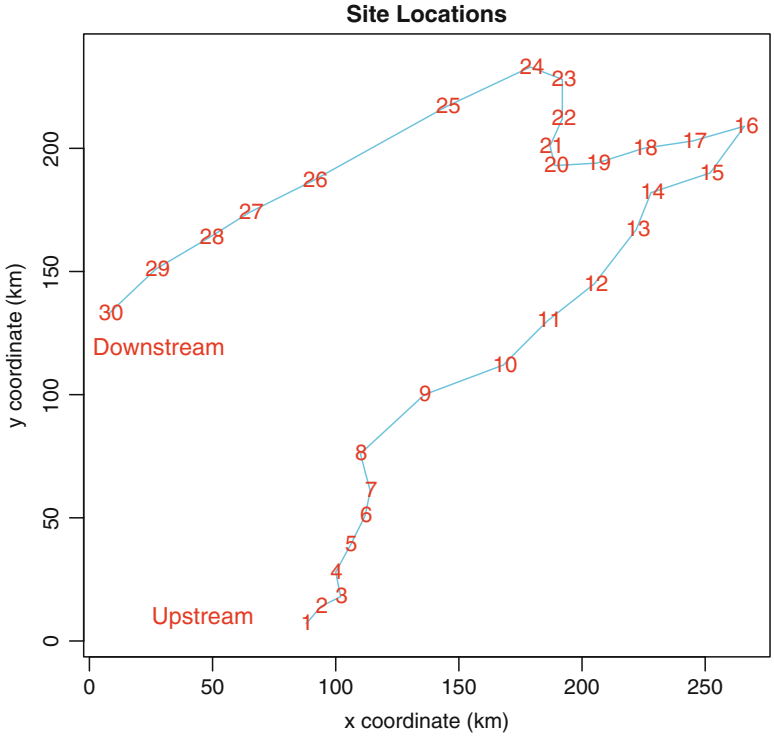

Fig. 2.2 Map of the 30 sampling sites along the Doubs River

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.*

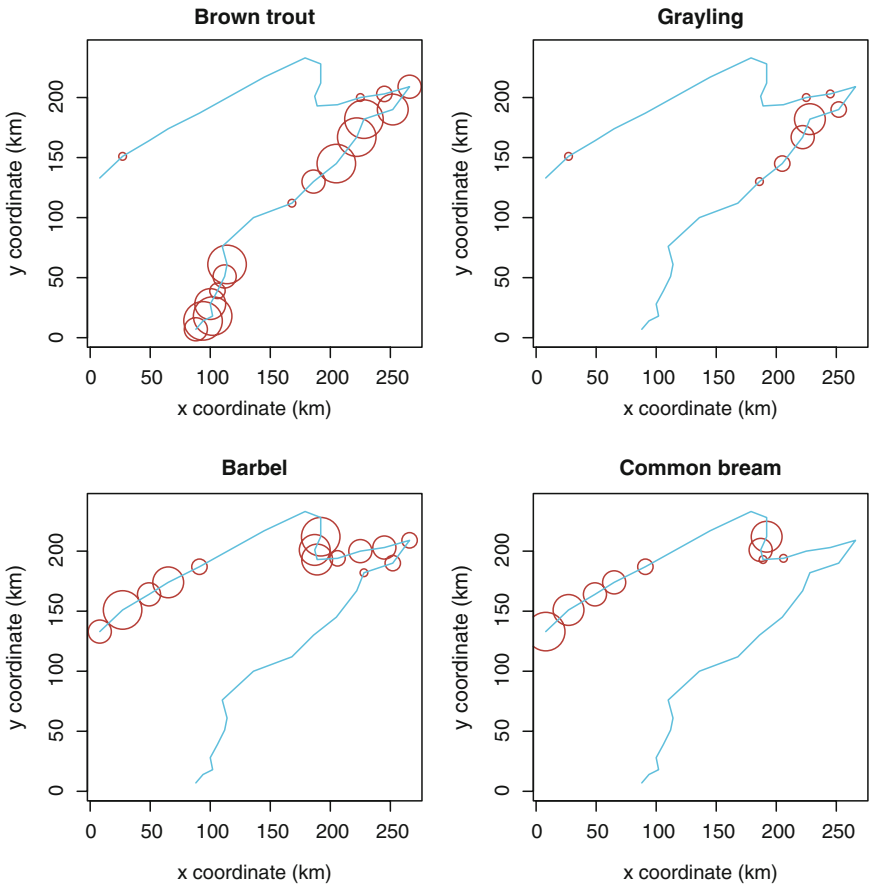

Fig. 2.3 Bubble maps of the abundance of four fish species

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.*

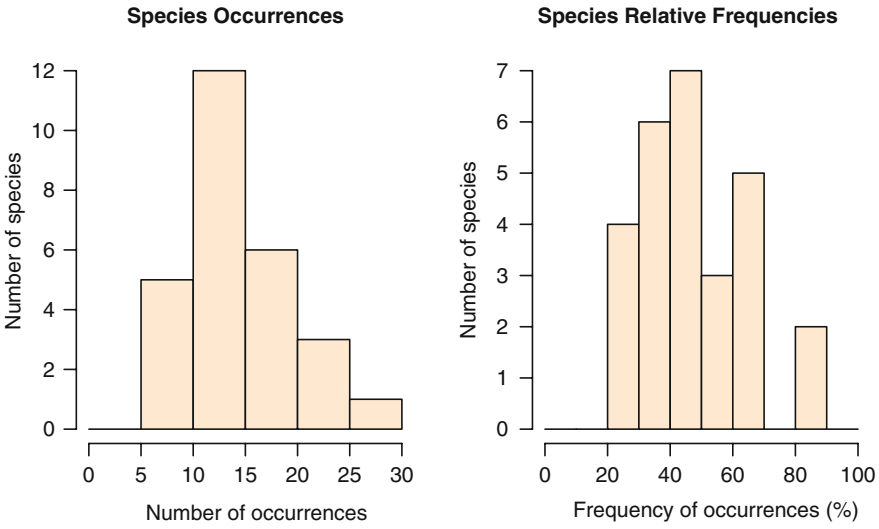

Fig. 2.4 Frequency histograms: species occurrences and relative frequencies in the 30 sites

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.*

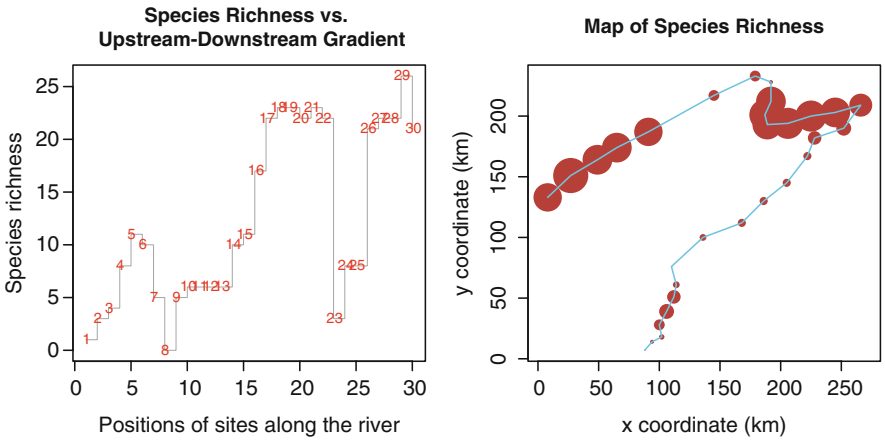

Fig. 2.5 Species richness along the river

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.*

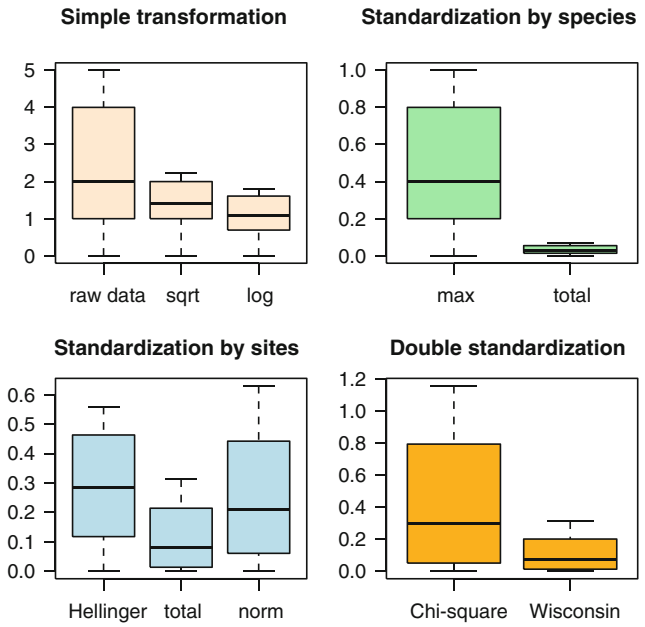

Fig. 2.6 Boxplots of transformed abundances of a common species, *Nemacheilus barbatulus* (stone loach)

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.*

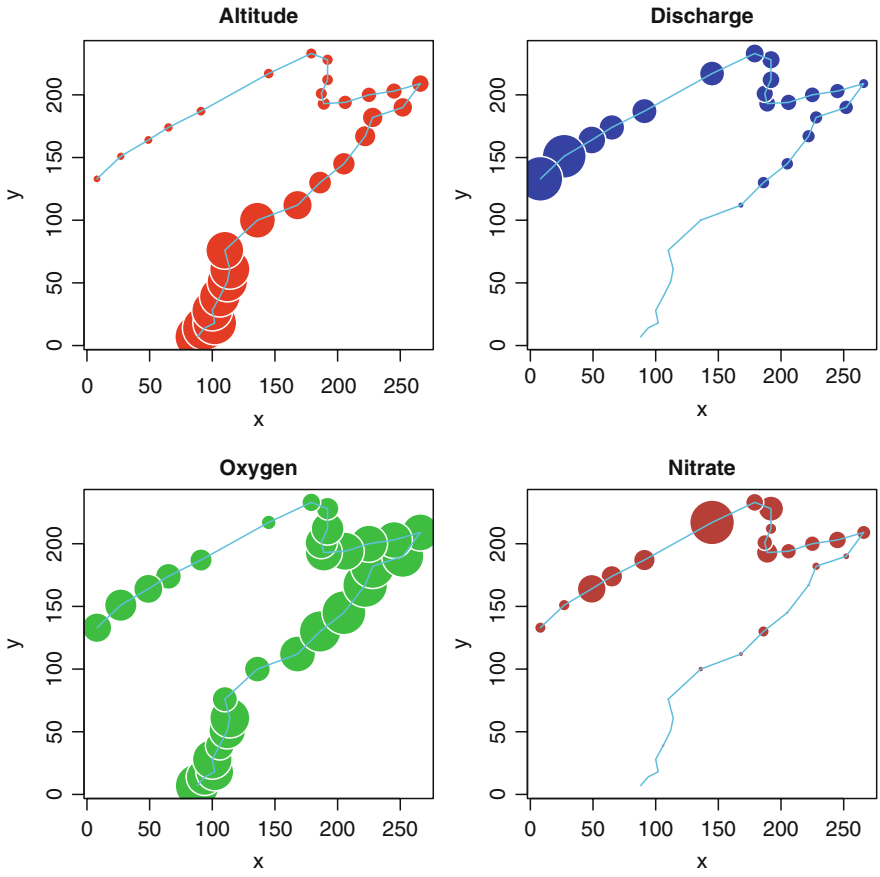

Fig. 2.7 Bubble maps of environmental variables

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")
```

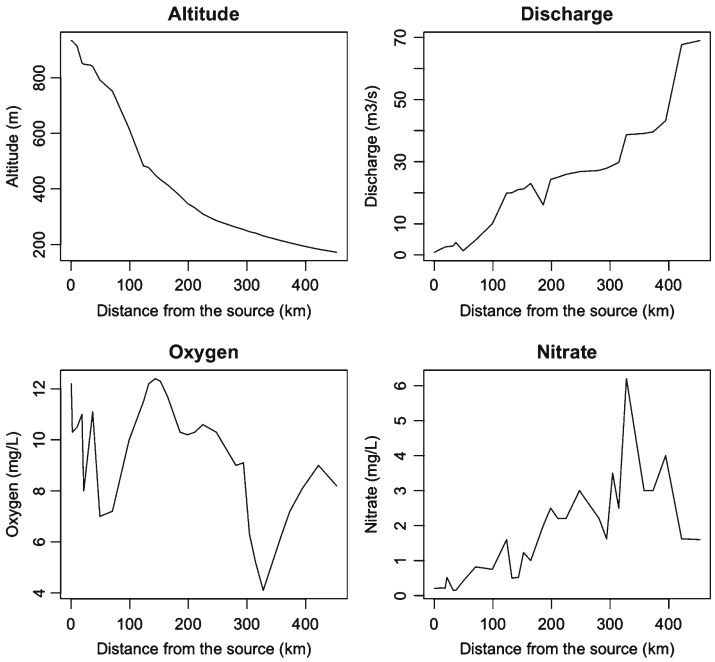

Fig. 2.8 Line plots of environmental variables

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).

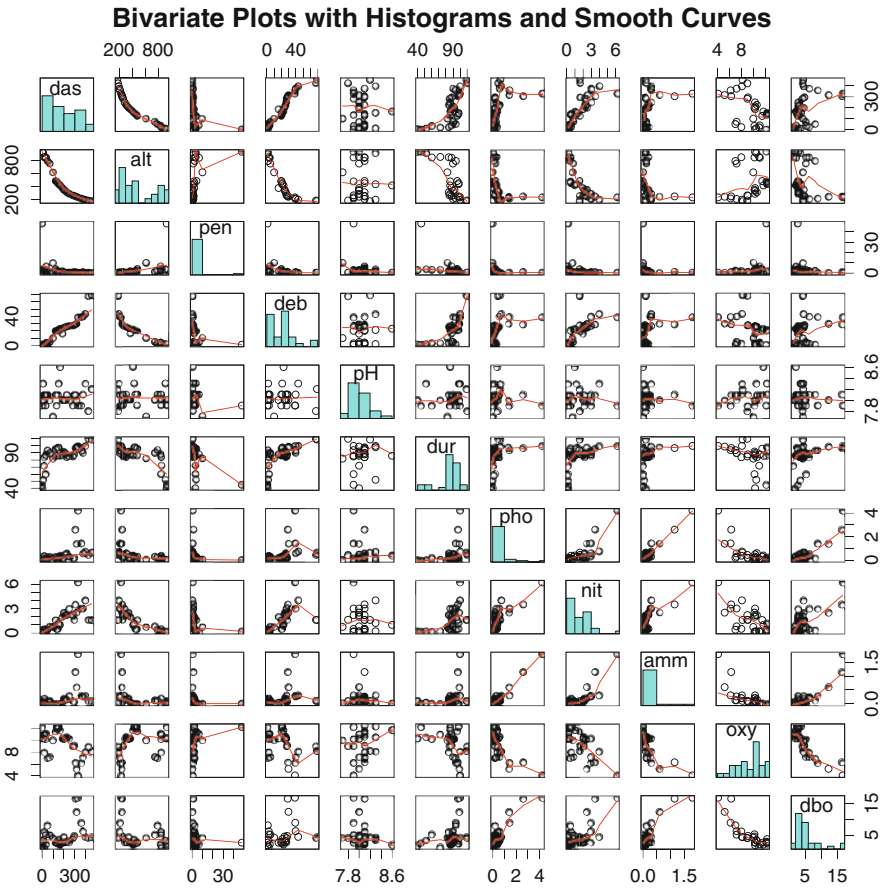

Fig. 2.9 Scatter plots between all pairs of environmental variables with LOWESS smoothers

```
# 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()`**.

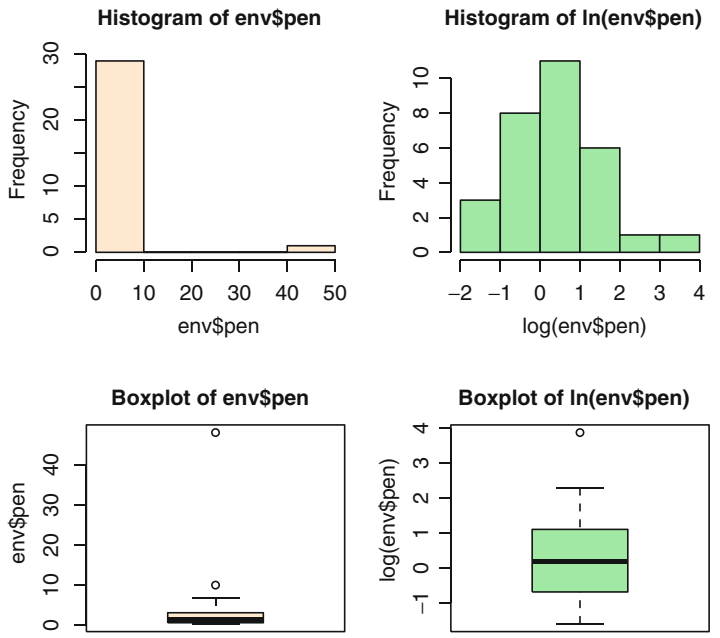

Fig. 2.10 Histograms and boxplots of the untransformed (*left*) and log-transformed `pen` variable (slope)

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.

http://www.springer.com/978-1-4419-7975-9

Numerical Ecology with R
Borcard, D.; Gillet, F.; Legendre, P.
2011, XII, 306 p., Softcover
ISBN: 978-1-4419-7975-9