adegraphics: An S4 Lattice-Based Package for the Representation of Multivariate Data

by Aurélie Siberchicot, Alice Julien-Laferrière, Anne-Béatrice Dufour, Jean Thioulouse and Stéphane Dray

Abstract The **ade4** package provides tools for multivariate analyses. Whereas new statistical methods have been added regularly in the package since its first release in 2002, the graphical functions, that are used to display the main outputs of an analysis, have not benefited from such enhancements. In this context, the **adegraphics** package, available on CRAN since 2015, is a complete reimplementation of the **ade4** graphical functionalities but with large improvements. The package uses the S4 object system (each graph is an object) and is based on the graphical framework provided by **lattice** and **grid**. We give a brief description of the package and illustrate some important functionalities to build elegant graphs.

Introduction

In many fields, data consists in tables containing measurements of several variables for a number of samples. In this context, multivariate analyses provide tools to summarize the main structures of multivariate data. After a dimension reduction step, these methods provide a graphical display of the primary relationships between variables and similarities between samples. Multivariate methods are routinely used in various fields including marketing, psychometry or ecology and have been implemented in several R packages (e.g. **vegan** (Oksanen et al., 2017), **MASS** (Venables and Ripley, 2002), **FactoMineR** (Lê et al., 2008), see the *Multivariate* Task View of CRAN). The **ade4** package (Dray and Dufour, 2007) is an alternative, distributed on CRAN since 2002, that implements more than 30 different methods, mainly oriented to the analysis of ecological data, and around 40 graphical functions to display the results of analyses.

During the last 15 years, a major effort was made in the implementation of new statistical methods in **ade4** but the graphical functionalities have not benefited from such improvements. At this time, graphs from **ade4** were black and white and lacked flexibility. It became increasingly difficult to customize the graphical outputs as the results of analyses are increasingly complex. In parallel, R has evolved and now proposes new paradigms of development (the S4 object system) and new graphical environments (**lattice** (Sarkar, 2008), `grid` (Murrell, 2005)). In this context, we started the development of a new add-on package: **adegraphics**. It is available on CRAN since 2015 and is designed to provide enhanced graphical functionalities for representing outputs of multivariate analysis, especially from **ade4**.

For users, **adegraphics** provides a flexible environment to produce, edit and easily manipulate graphs. To ensure continuity with the graphical functions of **ade4**, functions and parameters names have been largely preserved in **adegraphics**, so that regular users can easily migrate to the new package. Moreover, the graphical functions of **ade4** are not removed to preserve the functioning of other R packages and scripts that depend on **ade4**. The use of the new **adegraphics** functions is preferred and encouraged for future developments. Keeping the old version of the functions in **ade4** has some practical effects. In a classical case, when multivariate analyses are performed with **ade4**, it is imperative to load **ade4** before **adegraphics**. This solves the issues of conflicting names between functions by ensuring that the versions from **adegraphics** will be used.

```
> library(ade4)
> library(adegraphics)
```

The **adegraphics** package can be installed from CRAN. A development release is available on GitHub (`https://github.com/sdray/adegraphics`) and a detailled description of the package is given in the vignette available at `https://cran.r-project.org/web/packages/adegraphics/vignettes/adegraphics.html` or in any R session by typing `vignette("adegraphics")`. A reproducible version of the code in this paper is available online at `http://lbbe-shiny.univ-lyon1.fr/Reproducible_Research/TRJ.Siberchicot.2017/`.

This paper is divided in three parts: simple graphs, multiple graphs, and multivariate analysis graphs. The simple graphs part includes a presentation of basic elements and graphical parameters, with examples of graph manipulation and spatial representations. The multiple graphs part shows how to do automatic collections of graphs, and how to create and customize multiple graphs. The

multivariate analysis part shows an example of how the plot of a coinertia analysis can be customized and improved with **adegraphics**.

A simple example

We illustrate some principles and functionalities of **adegraphics** by considering the analysis of the `mafragh` data set available in **ade4**. It contains a phyto-ecological survey collected in an Algerian plain, called *La Mafragh* (de Bélair and Bencheikh-Lehocine, 1987; Pavoine et al., 2011).

```
> data("mafragh")
> names(mafragh)
```

```
[1] "xy"             "flo"            "neig"           "env"
[5] "partition"      "area"           "tre"            "traits"
[9] "nb"             "Spatial"        "spenames"       "Spatial.contour"
```

The aim of the original study (de Bélair and Bencheikh-Lehocine, 1987) was to propose an integrated management project of the Mafragh marshy coastal plain. `mafragh$flo` is a data frame giving the abundance of 56 plant species (columns) in 97 sites (rows) distributed in the plain. `mafragh$env` is a data frame with 97 rows and 11 columns describing the physico-chemical and topographic characteristics, and the granulometry of soil samples taken at the same sampling sites. `mafragh$partition` is a factor defined by de Bélair and Bencheikh-Lehocine (1987), classifying the 97 sites in 7 ecologically coherent regions according to observed vegetation units (ecoregions).

Basic elements and simple graphs

Classes, objects and calling functions

The **adegraphics** package uses object-oriented programming (OOP) in R: graphs are S4 objects that can be displayed or stored for later modification.

In **adegraphics**, there are two main parent classes of objects, one to store simple graphs (i.e. with only one represented information) called "ADEg" and the other, called "ADEgS", to store multiple graphs. The "ADEg" class has 5 sub-classes dedicated to different representation types: the "ADEg.T" class can be used to represent raw data such as distance matrices or contingency tables, the "ADEg.S2" class is associated with bidimensional data and can show factorial maps, the "ADEg.S1" class is used to represent a numeric score such as only one factorial axis in a unidimensional graph, the "ADEg.C1" class allows to represent unidimensional data associated with a supplementary information into two dimensions and the "ADEg.Tr" class is a peculiar class to represent data in a ternary plot. Each of these sub-classes itself has several child classes. The list of classes evolves and can be enriched to satisfy future developments. The **adegraphics** package provides user-friendly functions to create each type of graphical object. Some examples of these functions are listed in the Table 1 for each of the 5 sub-classes.

"ADEg" sub-classes	Example of sub-classes	Example of associated user functions
"ADEg.T"	"T.image", "T.value", ...	`table.image`, `table.value`, ...
"ADEg.S2"	"S2.class", "S2.corcircle", "S2.density", "S2.value", ...	`s.class`, `s.corcircle`, `s.density`, `s.value`, ...
"ADEg.S1"	"S1.boxplot", "S1.distri", "S1.match", ...	`s1d.boxplot`, `s1d.distri`, `s1d.match`, ...
"ADEg.C1"	"C1.curve", "C1.gauss", "C1.hist", ...	`s1d.curve`, `s1d.gauss`, `s1d.hist`, ...
"ADEg.Tr"	"Tr.class", "Tr.label", "Tr.match", "Tr.traject"	`triangle.class`, `triangle.label`, `triangle.match`, `triangle.traject`

Table 1: Examples of classes and associated graphical functions in `adegraphics`

In our example, a normed principal component analysis (PCA) is applied (using the `dudi.pca` function of **ade4**) on the `mafragh$env` data table. This table contains the measurements of 11 environmental variables for 97 sites. The first four axes are kept and the results are stored in object `pca1`.

```
> pca1 <- dudi.pca(mafragh$env, scale = TRUE, center = TRUE, scannf = FALSE, nf = 4)
```

The sites can be displayed on the factorial map formed by the first two axes. A graphical object is created (by the call to the `s.label` function) but not printed because of the argument `plot = FALSE`. Correlations between environmental variables and the first two PCA axes are represented on a correlation circle (with the user function `s.corcircle`, Figure 1). Both graphs (`g_sl1` and `g_sc1`) are stored in S4 objects of different sub-classes.

```
> g_sl1 <- s.label(pca1$li, plot = FALSE)
> class(g_sl1)

[1] "S2.label"
attr(,"package")
[1] "adegraphics"

> g_sc1 <- s.corcircle(pca1$co)
> class(g_sc1)

[1] "S2.corcircle"
attr(,"package")
[1] "adegraphics"
```

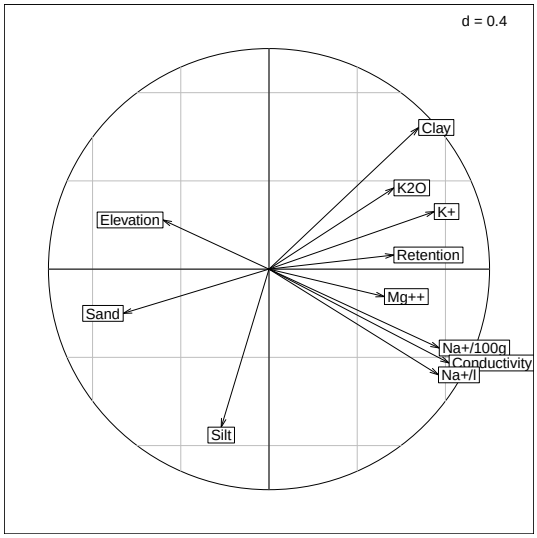

Figure 1: Default representation of the correlation circle of the 11 environmental variables on the first two PCA axes (`g_sc1`). The direction and length of arrows show correlations between variables and principal components. This graph helps to describe the Mafragh environment. The first axis is a salinity/elevation gradient with high salinity and low altitude on the right, opposed to high elevation and low salinity on the left. The second axis refines this gradient by opposing clay and potassium ions (up) to sodium ions and silts (down).

Each simple graphical object created with **adegraphics** (i.e. object belonging to class "ADEg") is defined by 8 attributes. These attributes are reachable by the @ operator and their names are given by the `slotNames` function.

```
> slotNames(g_sc1)

[1] "data"         "trellis.par"  "adeg.par"     "lattice.call" "g.args"
[6] "stats"        "s.misc"       "Call"
```

The `data` slot contains information about the data used in the plot and the `Call` slot contains the matched call.

```
> g_sc1@Call

s.corcircle(dfxy = pca1$co, xax = 1, yax = 2, labels = row.names(as.data.frame(pca1$co)),
    fullcircle = TRUE, facets = NULL, plot = TRUE, storeData = TRUE,
    add = FALSE, pos = -1)
```

Slots `s.misc` and `stats` store lists of internal parameters and internal preliminary computations. The `lattice.call` slot contains all the information required to create a "trellis" object (see the **lattice** package) associated to the **adegraphics** instruction. Note that an **adegraphics** object can be converted to a **lattice** one using the `gettrellis` function. The `trellis.par`, `adeg.par` and `g.args` slots manage several graphical parameters and are detailed in the next section.

Graphical parameters

A great flexibility is provided in **adegraphics** by the management of many graphical parameters. Parameter names are more intuitive and their number is greater than in **ade4**. There are 3 kinds of parameters in **adegraphics** user functions.

First, parameters implemented in **adegraphics** are itemized by the `adegpar` function (that works like the basic `par` function). When the graphical object is created, these parameters are stored in the `adeg.par` attribute. For example, the `adeg.par` attribute of the `g_sc1` object is yielded by `g_sc1@adeg.par` or by `getparameters(g_sc1,2)`. These parameters can be applied locally on a graph (as arguments to a user function) or globally in the working environment (using the `adegpar` function) and so applied on all graphs created thereafter.

Second, **lattice** parameters are managed in **adegraphics** and stored in the `trellis.par` attribute of the created object. For example, the `trellis.par` attribute of the `g_sc1` object is returned by `g_sc1@trellis.par` or by `getparameters(g_sc1,1)`. These parameters are itemized by the **lattice** function `trellis.par.get` and can also be applied locally or globally (see the `trellis.par.set` function of **lattice**).

Last, some general parameters are also managed like `xlim` and `ylim`, `xlab` and `ylab`, `main`, `scales`. These parameters are stored in the `g.args` attribute of the created object.

The **adegraphics** vignette explains more precisely how to manage these various graphical parameters and which one can be applied according to the class of the created object. The vignette appendix lists the correspondences between the graphical parameters used in **ade4** and their translation in **adegraphics**.

In the example below, the representation of environmental variables onto the first factorial map of `pca1` in the correlation circle (Figure 1) is customized: boxes around labels are removed and a subtitle is added. The background color of this new graph (called `g_sc2`) can be reached as the element `col` of the `pbackground` list of the `adeg.par` attribute. Here, the background color of `g_sc2` is `"white"`. Note that `g_sc2` is not printed right now (`plot = FALSE`).

```
> g_sc2 <- s.corcircle(pca1$co, plabels.boxes.draw = FALSE,
     psub.text = "Correlations of the environmental variables", plot = FALSE)
> g_sc2@adeg.par$pbackground$col

[1] "white"
```

Manipulating a simple graph

There are several methods in **adegraphics** to manipulate the graphical objects created and stored. For example, the `update` method allows to modify a posteriori some graphical parameters of a graph previously created with **adegraphics**, without recreating it. Besides, users can zoom in or out a stored graph thanks to the `zoom` method. Note that this function is only allowed for some sub-classes of **adegraphics**. Simply, the `plot`, `print` and `show` methods can be used to display a stored **adegraphics** object. Other functions and methods are described in the vignette of the **adegraphics** package.

In the example below, the grid and background color of the previously created `g_sc2` object are changed using the `update` method (Figure 2). As above, the background color parameter is reachable through the `adeg.par` slot of the `g_sc2` object; now it is `"grey90"`.

```
> update(g_sc2, pgrid.col = "white", pbackground.col = "grey90")
> g_sc2@adeg.par$pbackground$col

[1] "grey90"
```

Representation of spatial information

In **ade4**, classes were implemented to manage spatial information: the "area" class for geographic maps and the "neig" class to store spatial neighborhood objects. The **adegraphics** package abandoned

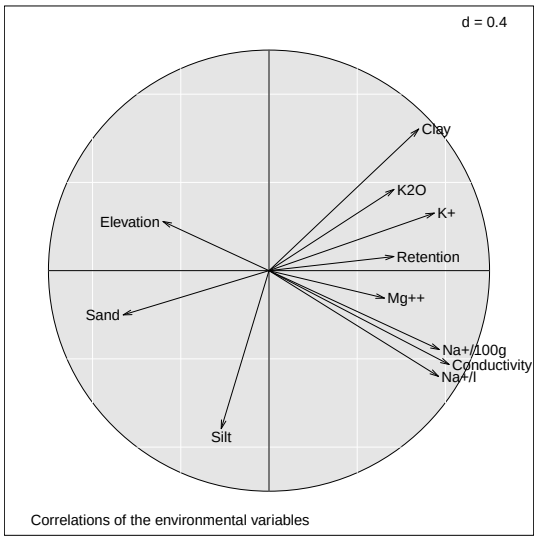

Figure 2: Customization of the g_sc2 object using the update method. Labels are more readable and the graph is highlighted by changing the colors and adding a title.

these outdated implementations and adopted more efficient classes. Maps are now considered as objects inherited from classes of the **sp** package (Pebesma and Bivand, 2005; Bivand et al., 2013b). "nb" and "listw" objects, from the **spdep** package (Bivand and Piras, 2015; Bivand et al., 2013a), are used to manage spatial neighborhood graphs. Note that only "ADEg.S2" objects (bidimensional plot of xy coordinates, implemented in s.* functions) of **adegraphics** support the representation of spatial objects by the use of arguments Sp and nb. See also the useful s.Spatial function to easily draw thematic maps. **adegraphics** provides specific parameters to manage the customization of spatial objects: the pSp parameters affect maps and pnb affects neighborhood.

In our example, a thematic map is simply obtained by displaying the score of sites on the first PCA axis (pca1$li[,1]) on a spatial map used as background. The s.value function is hereafter used to create the g_sv1 object (Figure 3, left) and takes as argument the mafragh$Spatial.contour object (an object of a class from the **sp** package) in the Sp parameter and a color palette in the col parameter. As many colors as classes of value are stored in a valuecolors vector created with the brewer.pal function of the **RColorBrewer** package (Neuwirth, 2014).

```
> class(mafragh$Spatial.contour)

[1] "SpatialPolygons"
attr(,"package")
[1] "sp"

> library(RColorBrewer)
> valuecolors <- rev(brewer.pal(6, "RdBu"))
> g_sv1 <- s.value(mafragh$xy, z = pca1$li[, 1], Sp = mafragh$Spatial.contour,
     method = "color", symbol = "circle", col = valuecolors, pgrid.draw = FALSE,
     ppoints.cex = 0.4)
```

A similar graph (called g_sp1) can easily be built (Figure 3, right) using the s.Spatial function and a "SpatialPolygonsDataFrame" object (from the **sp** package). The spobj object couples a "SpatialPolygons" object with the scores of sites on the first PCA axis.

```
> library(sp)
> spobj <- SpatialPolygonsDataFrame(Sr = mafragh$Spatial, data = pca1$li[,
     1, drop = FALSE], match.ID = FALSE)
> class(spobj)

[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"

> g_sp1 <- s.Spatial(spobj, col = valuecolors)
```

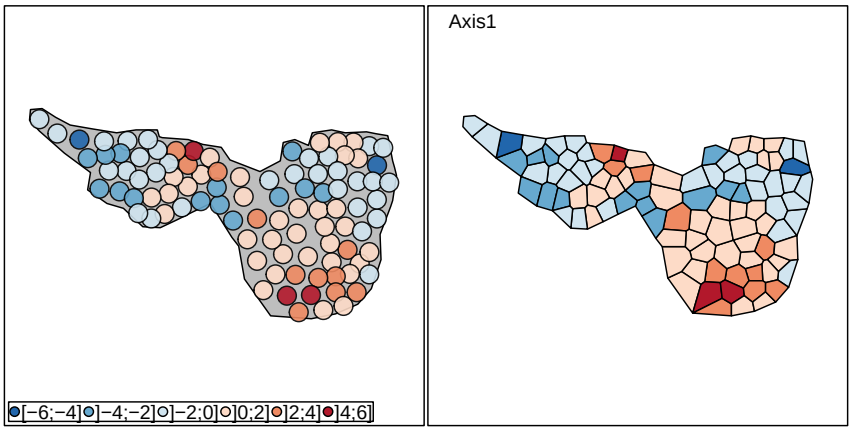

Figure 3: Thematic map: the scores on the first PCA axis are mapped in the geographical space. The `g_sv1` object (left) is created by a call to the `s.value` function and the `g_sp1` object (right) by the `s.Spatial` function. The color of sites is given by the value of scores on the first axis. The PCA first axis described by the correlation circle in Figure 2 helps interpret this figure: regions with high elevation (blue color, negative coordinates) are located at the extreme west and east, while the lowest points are located in the center of the map. Salinity (particularly potassium ions) is high in the southern region (red color, positive coordinates) because local conditions (clay) prevent the drainage of sea waters that accummulate there. Salinity (sodium ions) is also high in a particular region of the north because it is just next to the sea shore and receives a lot of sea water that stays trapped here.

Dealing with multiple graphs

The "ADEgS" class stores a collection of "ADEg", "ADEgS" and/or "trellis" (from the **lattice** package) objects, as a list of graphical objects. It is defined by 4 attributes, given by the `slotNames` function applied on a "ADEgS" object:

- `ADEglist` contains the list of graphs (i.e. objects) in the collection;
- `positions` is a matrix that contains the positions of subgraphs in the display;
- `add` is a matrix that contains the information about subgraphs superposition;
- `Call` contains the matching call.

"ADEgS" objects can be simply manipulated with methods associated to lists (i.e. `$`, `[]`, `[[]]`). The `names` function outputs the labels of subgraphs contained in an "ADEgS" object and the `length` function returns the number of subgraphs stored in the object.

We implemented different ways to generate collections of graphs. "ADEgS" objects can be created by a call to a simple function or built step-by-step using longer code. They can be customized in the same way as "ADEg" objects.

Automatic collections

Several user-friendly functionalities are implemented in **adegraphics** to facilitate the creation of collections of graphs, repeating (without using a for-loop command) a simple graph for different groups of individuals, variables or axes. The building of these collections is very simple for the user (definition of an argument in the call of a function) and leads to the creation of an "ADEgS" object.

For instance, as in the **ggplot2** package (Wickham, 2016), a `facets` argument is available to split a dataset into groups of individuals and to produce one graph per group. The `g_sv2` object (Figure 4) displays the scores of sites of the first PCA axis (as in the `g_sv1` object, left of Figure 3) for each of the 7 ecoregions, called `C1`, . . . ,`C7`, defined in `magragh$partition`.

```
> g_sv2 <- s.value(mafragh$xy, z = pca1$li[, 1], facets = mafragh$partition,
     Sp = mafragh$Spatial.contour, method = "color", symbol = "circle",
     col = valuecolors, pgrid.draw = FALSE, ppoints.cex = 0.4)
```

The `g_sv2` object is an "ADEgS" containing 7 subgraphs belonging to the "S2.value" class. The subgraphs names are given by the `names` function and are equal to the level names of `mafragh$partition`. Combined with the `$` symbol, these names can be used to select only one subgraph to the "ADEgS" object. For example, `g_sv2$C1` is the extraction of the `C1` subgraph of `g_sv2`.

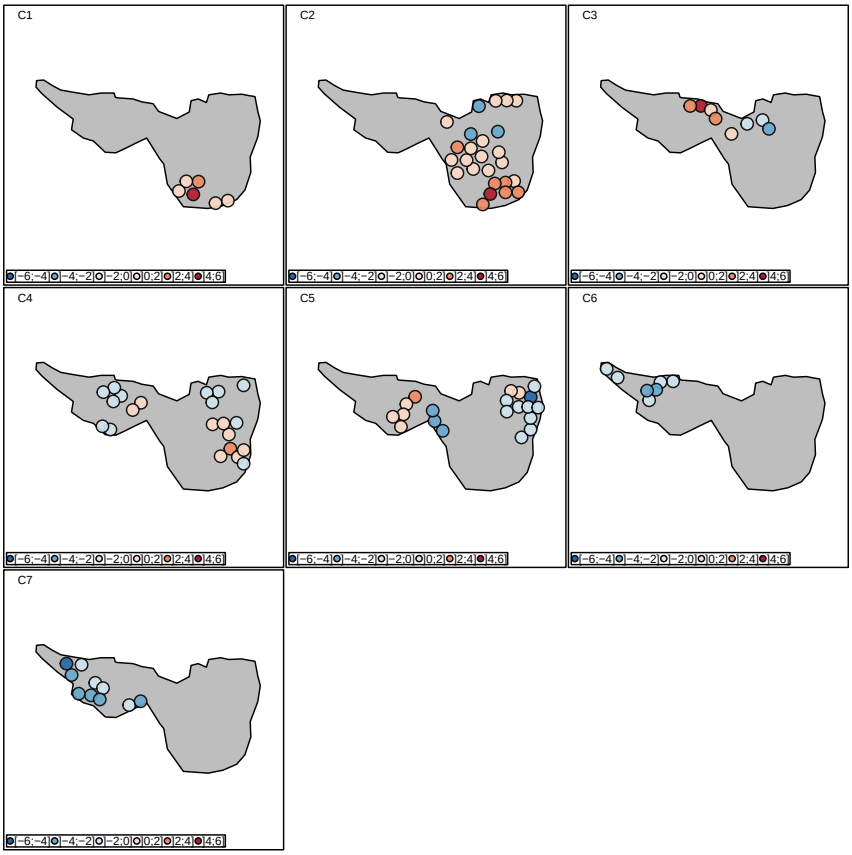

Figure 4: Automatic creation of an "ADEgS" object using the `facets` argument. The `g_sv2` object is created by only a call to the `s.value` function. Seven thematic maps (one for each ecoregion) are displayed to facilitate the read of PCA results. These 7 ecoregions were defined by de Bélair and Bencheikh-Lehocine (1987) from a correspondence analysis of the plant species data table (`mafragh$flo`) and phytosociological interpretations. Each ecoregion takes into account the distribution of plant species of various floristic groups, but also pedological characteristics of sampling sites, and geographical areas. For example, ecoregion `C1` is defined by 6 pedological sampling sites (numbered `43`, `44`, `45`, `46`, `49` and `51` in `g_sv2$C1@data$dfxy`) and the *Bolboschoenus maritimus* and *Schoenoplectus litoralis* plant communities (high abundances in `mafragh$flo[c(43, 44, 45, 46, 49, 51), ]`).

```
> class(g_sv2)

[1] "ADEgS"
attr(,"package")
[1] "adegraphics"

> names(g_sv2)

[1] "C1" "C2" "C3" "C4" "C5" "C6" "C7"

> class(g_sv2$C1)

[1] "S2.value"
attr(,"package")
[1] "adegraphics"
```

On the other hand, multiple graphs can also be produced by splitting graphs by columns. This is achieved when a data frame with several variables is given as an argument to a function that usually requires a vector. For instance, the abundance of four focused species is easily displayed along the `mafragh` sites. The `g_sv3` object (Figure 5) is created with the `s.value` function, using the `mafragh$flo[, selectedspecies]` data frame as `z` parameter. A line surrounding the entire `mafragh` region (contained in the `mafragh$Spatial.contour` object) is added thanks to the `Sp` argument.

```
> selectedspecies <- c(9, 12, 31, 34)
> floselectedspecies <- mafragh$flo[, selectedspecies]
```

```
> colnames(floselectedspecies) <- mafragh$spenames$code[selectedspecies]
> colnames(floselectedspecies)

[1] "Juma" "Scli" "Boof" "Plco"

> g_sv3 <- s.value(mafragh$xy, z = floselectedspecies, symbol = "circle",
     centerpar = list(cex = 0.1), ppoints.cex = 0.7, pgrid.draw = FALSE, psub.cex = 2,
     porigin.draw = FALSE, plegend.drawK = FALSE, Sp = mafragh$Spatial.contour)
```

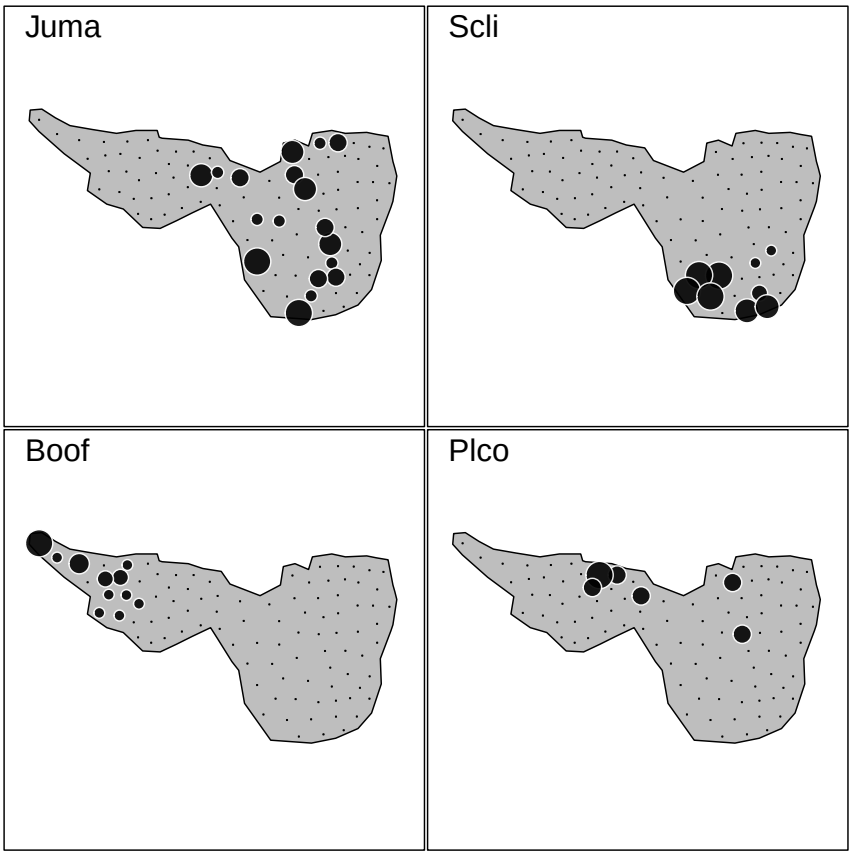

Figure 5: Automatic creation of an "ADEgS" object. The `g_sv3` object is created by only a call to the `s.value` function using a multivariate argument `z`. Four maps of abundance are produced, one for each of four species of interest: *Juncus maritimus* (`Juma`), *Schoenoplectus litoralis* (`Scli`), *Borago officinalis* (`Boof`) and *Plantago coronopus* (`Plco`).

Step-by-step creation

"ADEgS" objects can also be created by the manipulation of several simple graphs. Several functions in **adegraphics** allow superposition, combination or insertion of two objects: `superpose`, `+`, `cbindADEg`, `rbindADEg` and `insert`.

Following our example, a Correspondence Analysis (CA, called `coa1`) is performed on the abundance of the 56 plant species of `mafragh` (contained in the `mafragh$flo` data table) with the `dudi.coa` function of the **ade4** package, displayed but not shown with the `s.label` function and stored in a `g_sl2` object. Note here the `plabel.optim` parameter set to `TRUE` which automatically optimize the label positions and remove label boxes.

```
> coa1 <- dudi.coa(mafragh$flo, scannf = FALSE, nf = 2)
> g_sl2 <- s.label(coa1$co, labels = mafragh$spenames$code, plabel.optim = TRUE,
     plot = FALSE)
```

To help interpret the results of this CA, three among four maps of species abundance created in the `g_sv3` object can be inserted, one at a time, close to the matching species point on the `g_sl2` object. The `insert` function takes as parameters a first graph to insert, a second one in which to insert and a

posi argument to define the insertion position. Each of the three inserted subgraph is extrated from the g_sv3 object by the [[]] method. The final output called g_in1 is an object of "ADEgS" class (Figure 6).

```
> g_in1 <- insert(g_sv3[[1]], g_sl2, posi = c(0.06, 0.16), plot = FALSE)
> g_in1 <- insert(g_sv3[[2]], g_in1, posi = c(0.15, 0.75), plot = FALSE)
> g_in1 <- insert(g_sv3[[3]], g_in1, posi = c(0.57, 0.77))
> class(g_in1)

[1] "ADEgS"
attr(,"package")
[1] "adegraphics"
```

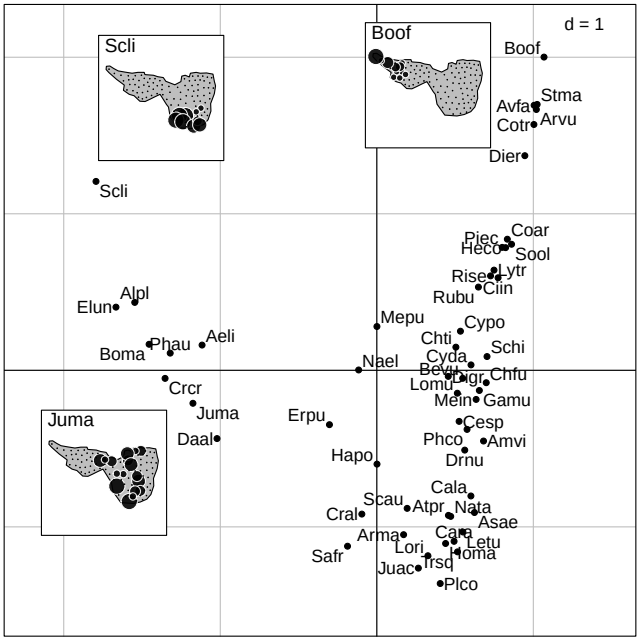

Figure 6: Creation of an "ADEgS" by the manipulation of four "ADEg" objects. This display can be produced by the insert or ADEgS function. Three geographical maps are superimposed on the first factor map of the CA of the plant species table. The spatial distribution of *Juncus maritimus* (Juma), *Schoenoplectus litoralis* (Scli) and *Borago officinalis* (Boof) are very different, and the CA scores clearly take these differences into account, even though spatial coordinates of sampling sites are not used in the analysis. They correspond to ecoregions C1, C2 and C6, respectively.

This object has four elements (i.e. graphs). As for a standard list, the function names can be used to modify labels of each g_in1 subgraph:

```
> names(g_in1)

[1] "g1"  "g2"  "X"   "X.1"

> names(g_in1) <- c("CA", "Juma", "Scli", "Boof")
> names(g_in1)

[1] "CA"   "Juma" "Scli" "Boof"
```

The insert function is a shortcut that calls a very flexible function designed to generate "ADEgS" objects. The general ADEgS function allows to arrange multiple graphs. It takes as arguments (i) a list of several "ADEg", "ADEgS" and/or "trellis" objects and (ii) information about their layout.

Below, the g_in2 object, created by the ADEgS function, is strictly identical to g_in1. The positions of each of the four subgraphs is yielded by the positions attribute of the g_in1 object.

```
> g_in1@positions

     [,1] [,2] [,3] [,4]
     0.00 0.00 1.00 1.00
```

```
positions 0.06 0.16 0.26 0.36
positions 0.15 0.75 0.35 0.95
positions 0.57 0.77 0.77 0.97
```

```
> g_in2 <- ADEgS(list(CA = g_sl2, Juma = g_sv3[[1]], Scli = g_sv3[[2]],
     Boof = g_sv3[[3]]), positions = rbind(c(0, 0, 1, 1), c(0.06,
     0.16, 0.26, 0.36), c(0.15, 0.75, 0.35, 0.95), c(0.57, 0.77,
     0.77, 0.97)), plot = FALSE)
```

Customizing an "ADEgS"

Like simple graphs ("ADEg"), the multiple graphs generated with **adegraphics** can be customized during or after the creation of the object, thanks to the update function. It is possible to apply changes to all or only one subgraph contained in the "ADEgS". To modify a graphical parameter for all subgraphs of the "ADEgS", the syntax is the same as for a simple "ADEg". To modify a graphical parameter for only a given subgraph, the parameter name must be preceded by the name of the subgraph, separated by a dot. This supposes that the subgraph names (available by a call to the names function) is known.

For example, the background color of the g_in1 object can be updated for all subgraphs:

```
> update(g_in1, pbackground.col = "grey90")
```

or only for the one called CA:

```
> update(g_in1, CA.pbackground.col = "grey90")
```

The "ADEgS" created hereafter by the ADEgS function is a good example how to manipulate "ADEg" objects and deal with **adegraphics** graphical parameters. The g_adegs1 object is a personal graph (Figure 7) to explore pca1 results. It arranges four "ADEg" objects in a personal layout.

First, default global graphical parameters are stored in a oldadegpar object in order to be restored at the end by the adegpar function.

```
> oldadegpar <- adegpar()
```

Two graphs called g_gau1d and g_lab1d are created to represent the scores of ecoregions and environmental variables on the first pca1 axis. Parameters about unidimensional displays, affecting this two graphs, are globally modified by the call of the adegpar function.

```
> adegpar(p1d = list(horizontal = FALSE, rug.tck = 1, margin = 0.07))
> g_gau1d <- s1d.gauss(pca1$li[, 1], fac = mafragh$partition, col = c(1:6, 8),
     p1d.reverse = TRUE, p1d.rug.margin = 0.1, plabels.cex = 2, plot = FALSE)
> g_lab1d <- s1d.label(pca1$co[, 1], labels = rownames(pca1$co), plabels.cex = 2,
     plot = FALSE)
```

Then the thematic map of the scores of sites on the first pca1 axis (g_sv1, Figure 3 at left) is kept back.

Finally the graph of the pca1 eigenvalues is created (called g_eig) with the relevant plotEig function of **adegraphics**. Here, only the first axis is kept to interpret the results (see the black bar).

```
> g_eig <- plotEig(pca1$eig, xax = 1, yax = 1, nf = 1, pbackground.box = TRUE,
     plot = FALSE)
```

The g_adegs1 can now be created, arranging the four graphs previously stored.

```
> g_adegs1 <- ADEgS(list(g_gau1d, g_lab1d, g_sv1, g_eig),
     layout = matrix(c(1, 2, 3, 1, 2, 4), nrow = 2, byrow = TRUE), plot = FALSE)
```

After the "ADEgS" creation, some graphical parameters can be a posteriori updated: the grid is removed on all graphs, a title is added to the third (called g3 in the g_adegs1 object) and fourth (called g4 in the g_adegs1 object) graphs and these titles are enlarged.

```
> names(g_adegs1)
```

```
[1] "g1" "g2" "g3" "g4"
```

```
> update(g_adegs1, pgrid.draw = FALSE, psub.cex = 1.6,
     g3.psub = list(text = "Map of scores of the first PCA axis"),
     g4.psub = list(text = "PCA eigenvalues"))
> adegpar(oldadegpar)
```

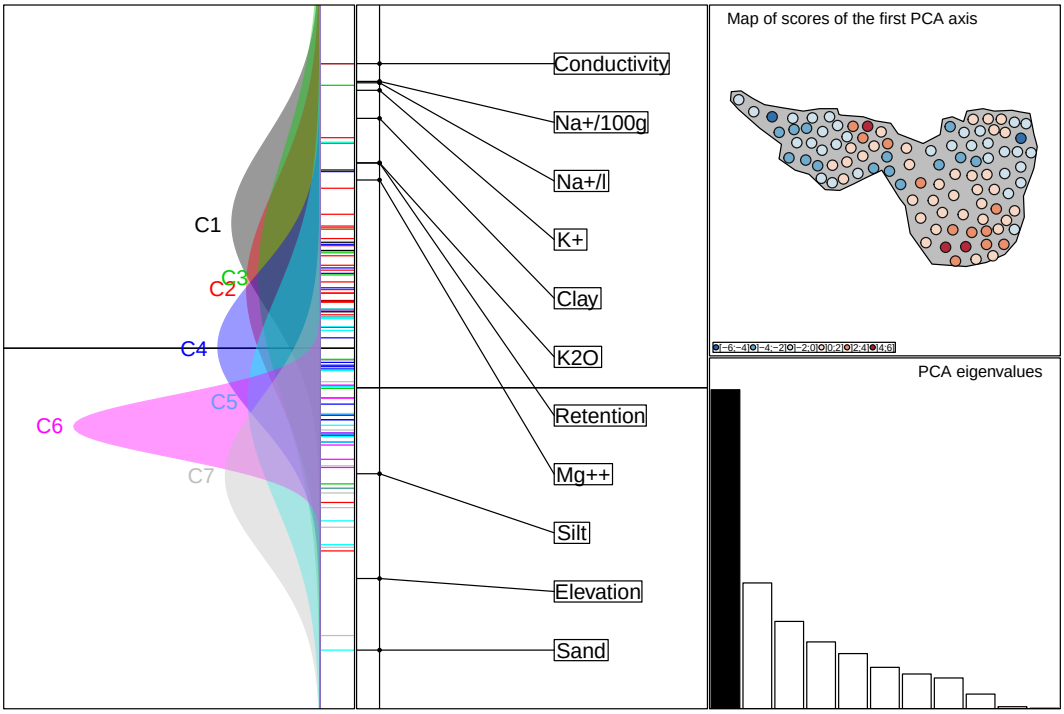

Figure 7: Creation of an "ADEgS" by the manipulation of four "ADEg" objects. Some graphical parameters are globally set before creating the graphs, others are defined when each subgraph is created, and others are updated after the final creation. The left part of the figure is the one-dimensional biplot of the PCA first axis. Gauss curves on the left show the distribution of sampling site PCA scores for the seven ecoregions. The labels on the right show the physico-chemical characteristics of each ecoregion, as evidenced by the PCA. For example, the sampling sites of ecoregion `C1` are characterized by a high conductivity and high levels of salinity, while sites of ecoregions C5, C6 and C7 are higher and more sandy. The top right map recalls that this first axis has a strong spatial structure (see legend to Figure 1), and the bottom right barplot shows that this corresponds to a large part (41%) of the total inertia.

Multivariate analysis outputs

All functions formerly available in **ade4** to display the outputs of multivariate analyses have been reimplemented in **adegraphics**. Most of them return an "ADEgS" object.

In our example, the PCA on environmental variables is now applied with the CA weights (`coa1$lw`) and stored in `pca2`. Then, a coinertia analysis is built to identify relationships between environmental variables (`pca2`) and species distributions (`coa1`). The results are stored in a `coi1` object and are graphically summarized by a call to the `plot` method.

```
> pca2 <- dudi.pca(mafragh$env, row.w = coa1$lw, scannf = FALSE, nf = 2)
> coi1 <- coinertia(coa1, pca2, scannf = FALSE, nf = 3)
> g_coi1 <- plot(coi1)
```

The resulting graphical object (`g_coi1`, Figure 8) is an "ADEgS" containing 6 subgraphs displaying the main outputs required to interpret the results.

```
> length(g_coi1)

[1] 6
```

In **ade4**, the graphical outputs provided by the `plot` function were initially designed to obtain a quick view on the main outputs. However, the number of graphs and their relatively small size did not allow a deep interpretation of the results. Hence, it was often required to write some new R code to redraw only the useful information in a publication-ready format. In **adegraphics**, the process is simplified because results are produced as "ADEgS" objects that can be manipulated as standard lists. Each graph is an element of this list and can be easily extracted using the names of elements (for the `$` operator) or their index in the list (for `[[]]`). The way a graph is created with **adegraphics** is reachable with the `Call` attribute of this object.

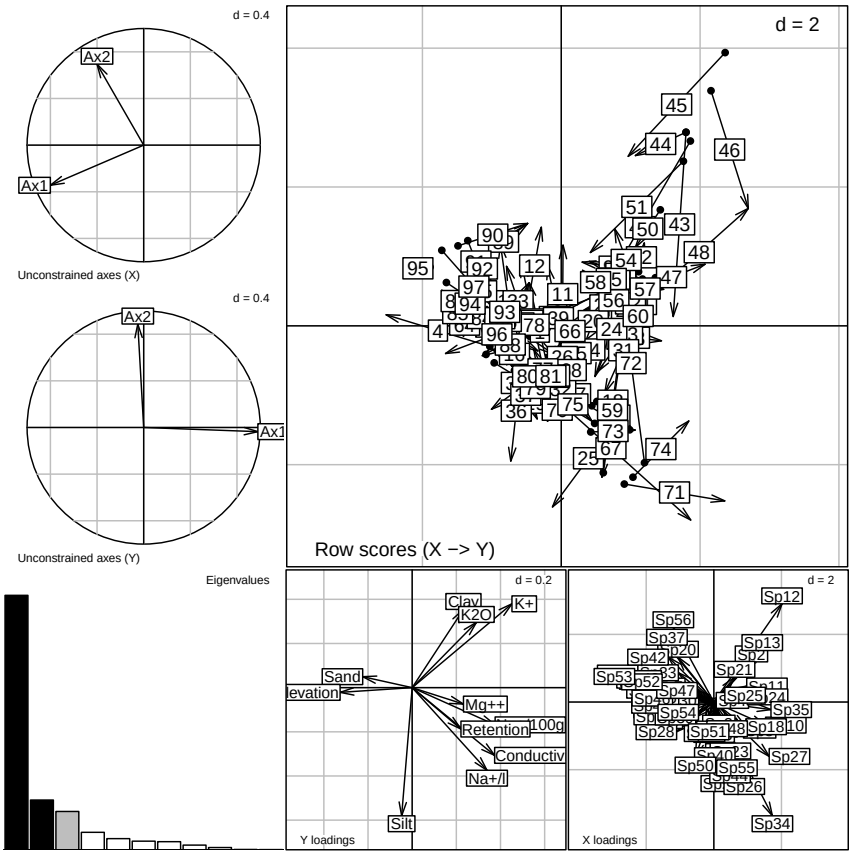

Figure 8: Default representation of a coinertia analysis. The `g_coi1` object is a collection of 6 graphs produced by the `plot.coinertia` method implemented in **adegraphics**.

```
> names(g_coi1)

[1] "Xax"       "Yax"       "eig"       "XYmatch"   "Yloadings" "Xloadings"

> g_coi1$XYmatch      # the same as g_coi1[[4]]

> g_coi1@Call

plot.coinertia(x = coi1)

> g_coi1$XYmatch@Call

s.match(dfxy1 = coi1$mX, dfxy2 = coi1$mY, xax = 1, yax = 2, plot = FALSE,
    storeData = TRUE, pos = -3, psub = list(text = "Row scores (X -> Y)"),
    labels = row.names(as.data.frame(coi1$mX)), arrows = TRUE,
    facets = NULL, add = FALSE)
```

In some subgraphs of Figure 8, information is not easy to read and not necessarily relevant. Thanks to the graphical facilities developped in **adegraphics**, a new multiple graph can be created to better explore coinertia results.

First, to improve the top-right subgraph (`g_coi1$XYmatch`) which is unreadable, a scatter plot is created (`g_scl`) to represent the scores of sites on the first factorial map of the coinertia, not for each sites (as in `g_coi1$XYmatch`) but grouped and colored by ecoregion. To build the new graph, it is usefull to know how `g_coi1$XYmatch` was created thanks to its `Call` attribute (see above). Then the eigenvalues barplot of the `coi1` coinertia (the subgraph called `g_coi1$eig`, located at the bottom left of `g_coi1`, in Figure 8) is inserted at the topleft of `g_scl`.

```
> g_scl1 <- s.class(coi1$mX, fac = mafragh$partition, ellipseSize = 0, chullSize = 0,
     starSize = 0.5, ppoints.cex = 0, col = c(1:6, 8), plot = FALSE)
> g_scl2 <- s.class(coi1$mY, fac = mafragh$partition, ellipseSize = 0, chullSize = 0,
     starSize = 0.5, ppoints.cex = 0, col = c(1:6, 8), plot = FALSE)
```

```
> g_scs1 <- superpose(g_scl1, g_scl2)
> g_sm1 <- s.match(g_scl1@stats$means, g_scl2@stats$means, plabels.cex = 0,
    plines.lwd = 2, plot = FALSE)
> g_scl <- superpose(g_scs1, g_sm1)
> g_scl <- insert(g_coi1$eig, g_scl, posi = "topleft", ratio = 0.3, plot = FALSE)
```

Next, the `g_coi1$Yloadings` subgraph is stored in a new `g_yload` object and then updated: boxes around labels and grid are removed; the subtitle is enlarged.

```
> g_yload <- g_coi1$Yloadings
> update(g_yload, plabels.box.draw = FALSE, pgrid.draw = FALSE, psub.cex = 1.2,
    plot = FALSE)
```

Then a `g_sl3` object is inspired by the bottom-right subgraph (`g_coi1$Xloadings`) of `g_coi1`: a `s.label` function is used instead of the `s.arrow` one, labels are extracted from the `mafragh$spenames$code` vector, labels positions are optimized, points and grid are removed.

```
> g_sl3 <- s.label(dfxy = coi1$c1, psub = list(text = "X loadings"),
    labels = mafragh$spenames$code, plabels.optim = TRUE, plabels.cex = 1.2,
    psub.cex = 1.2, ppoints.cex = 0, pgrid.draw = FALSE, plot = FALSE)
```

The `g_adegs2` object is at last created with the `ADEgS` function.

```
> g_adegs2 <- ADEgS(list(g_scl, g_yload, g_sl3),
    layout = list(mat = matrix(c(1, 1, 2, 1, 1, 3), nrow = 2, byrow = TRUE)))
```

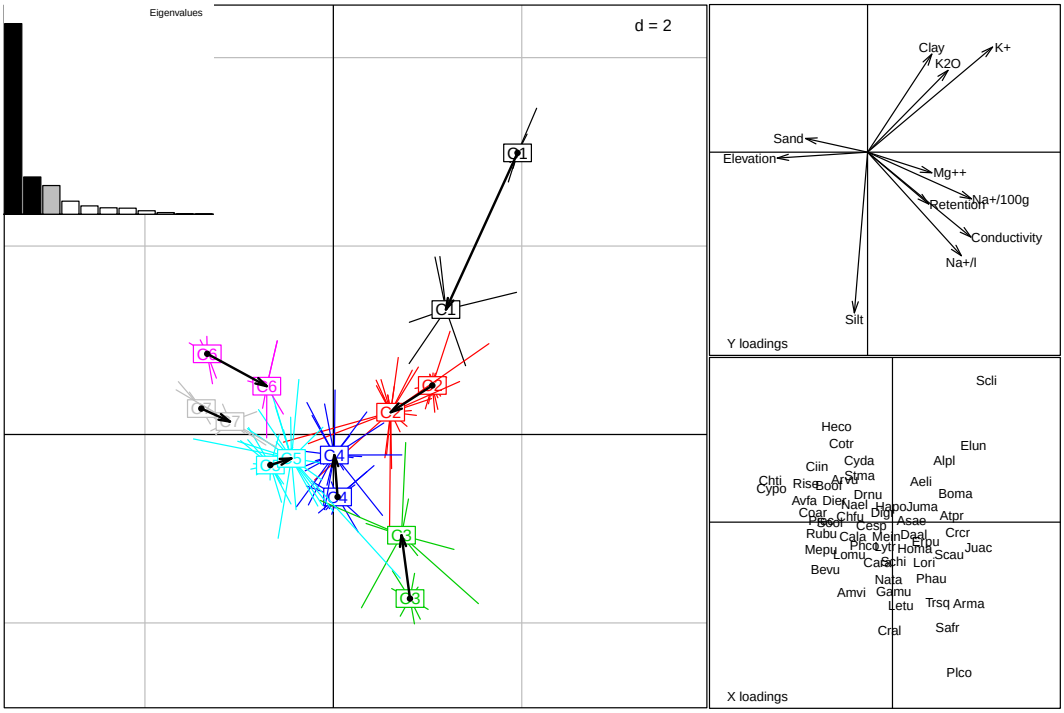

Figure 9: A customized coinertia analysis. The `g_adegs2` object is an improvement to the default `g_coi1` one. Each graphical parameter can be customized before, during and/or after the object creation and the "ADEgS" building becomes easy. This figure helps interpret the relationships between plant species and environmental parameters. For example, *Schoenoplectus Litoralis* (`Scli`) is found preferentially in sites with high clay and potassium rates. On the opposite, *Plantago coronopus* (`Plco`) is found in sites with high levels of sodium ions and high conductivity. *Cynosurus polybracteatus* (`Cypo`) and *Chrozophora tinctoria* (`Chti`) are found in sandy sites with high elevation above sea level. The discrepancy between plant and environmental parameters in the 7 ecoregions is shown in the left graph. For each ecoregion, the coordinates of sites are linked to their gravity center, forming an irregular star. The gravity centers of site coordinates from the plants table and from the environmental parameters table are linked by an arrow. The length of this arrow is therefore representative of the discrepancy between plants and environmental parameters. Regions `C1` and `C6` have to the longest arrows, which means that the discrepancy is higher in these three ecoregions.

Conclusions

The **adegraphics** package is a complete reimplementation of **ade4** graphical functionalities with large improvements. The structure of this new package is quite rigorous (formalism provided by S4 classes) but very flexible and easily extendable to support new features. The graphical engine of the **lattice** package allows to produce high quality graphs for data exploration and publication. This **lattice**-based implementation provides an efficient and highly-customizable object-based environment to build graphs and it also facilitates the connections between packages. The "ADEgS" class allows to integrate simple "trellis" objects from the **lattice** package and thus offers the possibility to integrate, in the same graphical object, various graphs created by different packages.

This code is executed with R 3.4.2, **ade4** 1.7-8 and **adegraphics** 1.0-8. Other examples are available in the vignette and in the help pages of the package.

Bibliography

R. S. Bivand and G. Piras. Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18):1–36, 2015. URL https://doi.org/10.18637/jss.v063.i18. [p5]

R. S. Bivand, J. Hauke, and T. Kossowski. Computing the Jacobian in Gaussian Spatial Autoregressive Models: An Illustrated Comparison of Available Methods. *Geographical Analysis*, 45(2):150–179, 2013a. URL https://doi.org/10.1111/gean.12008. [p5]

R. S. Bivand, E. J. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R, Second Edition*. Springer-Verlag, 2013b. URL https://doi.org/10.1007/978-1-4614-7618-4. [p5]

G. de Bélair and M. Bencheikh-Lehocine. Composition et déterminisme de la végétation d'une plaine côtière marécageuse : La Mafragh (Annaba, Algérie). *Bulletin d'écologie*, 18(4):393–407, 1987. [p2, 7]

S. Dray and A.-B. Dufour. The ade4 Package: Implementing the Duality Diagram for Ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007. URL https://doi.org/10.18637/jss.v022.i04. [p1]

S. Lê, J. Josse, and F. Husson. FactoMineR: An R Package for Multivariate Analysis. *Journal of Statistical Software*, 25(1):1–18, 2008. URL https://doi.org/10.18637/jss.v025.i01. [p1]

P. Murrell. *R Graphics*. Chapman & Hall / CRC Press, 2005. URL http://www.e-reading.club/bookreader.php/137370/C486x_APPb.pdf. [p1]

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-2. [p5]

J. Oksanen, F. G. Blanchet, M. Friendly, R. Kindt, P. Legendre, D. McGlinn, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, E. Szoecs, and H. Wagner. *Vegan: Community Ecology Package*, 2017. URL https://CRAN.R-project.org/package=vegan. R package version 2.4-4. [p1]

S. Pavoine, E. Vela, S. Gachet, G. de Bélair, and M. B. Bonsall. Linking patterns in phylogeny, traits, abiotic variables and space: a novel approach to linking environmental filtering and plant community assembly. *Journal of Ecology*, 99(1):165–175, 2011. URL https://doi.org/10.1111/j.1365-2745.2010.01743.x. [p2]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, 2005. URL https://www.r-project.org/doc/Rnews/Rnews_2005-2.pdf. [p5]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York, 2008. URL https://doi.org/10.1007/978-0-387-75969-2. [p1]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL https://doi.org/10.1007/978-0-387-21706-2. [p1]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, New York, 2nd edition, 2016. URL https://doi.org/10.1007/978-3-319-24277-4. [p6]

Aurélie Siberchicot
Univ Lyon, Université Claude Bernard Lyon 1, CNRS,
Laboratoire de Biométrie et Biologie Evolutive,
F-69100, Villeurbanne, France.

ORCiD: 0000-0002-7638-8318
aurelie.siberchicot@univ-lyon1.fr

Alice Julien-Laferrière
Univ Lyon, Université Claude Bernard Lyon 1, CNRS,
Laboratoire de Biométrie et Biologie Evolutive,
F-69100, Villeurbanne, France.
alice.julien.laferriere@gmail.com

Anne-Béatrice Dufour
Univ Lyon, Université Claude Bernard Lyon 1, CNRS,
Laboratoire de Biométrie et Biologie Evolutive,
F-69100, Villeurbanne, France.
ORCiD: 0000-0002-9339-4293
anne-beatrice.dufour@univ-lyon1.fr

Jean Thioulouse
Univ Lyon, Université Claude Bernard Lyon 1, CNRS,
Laboratoire de Biométrie et Biologie Evolutive,
F-69100, Villeurbanne, France.
ORCiD: 0000-0001-7664-0598
jean.thioulouse@univ-lyon1.fr

Stéphane Dray
Univ Lyon, Université Claude Bernard Lyon 1, CNRS,
Laboratoire de Biométrie et Biologie Evolutive,
F-69100, Villeurbanne, France.
ORCiD: 0000-0003-0153-1105
stephane.dray@univ-lyon1.fr