Formation MARBEC, Sète, 20-21 janvier 2020

Multivariate analysis with R

One table analyses

Monique Simier, IRD, UMR MARBEC, Pôle Modélisation

This R course deals with the multivariate methods allowing to study the structure of a two dimensional data table. We'll use for this purpose the functions from the **ade4 library**. We'll first talk about **ordination methods** such as **Principal Component Analysis** (1), **Correspondence Analysis** (2), **Multiple Correspondence Analysis** (3). Finally, we will describe the methods for **automatic classification**, hierarchical or not, through the functions **hclust()** and **kmeans()** available in the R base software (4).

Sommaire

1. Principal Component Analysis (PCA)

1.1. Introduction to PCA

Principal Component Analysis (**PCA**) is a multivariate analysis method designed for the analysis of large rectangular data table (of class data.frame), with **n individuals as rows** and **p variables as columns**. The variables must all be quantitative but they can be in different units. With the default "Normed" PCA, the data table is first standardized (all variables become of mean 0 and standard deviation 1). You can also choose to only center them (this is the "Centered" PCA), if the variables all have the same units. For the Normed PCA, the **correlation matrix** between variables is computed (covariance matrix if Centered PCA). Eigenvalues are computed from the correlation or covariance matrix. Axes, corresponding to the eigenvalues, are built to project individuals and variables. The number of axes corresponds to the number of variables p. The total information in the dataset (also called **inertia**) is distributed on the p axes. The first one is considered as the best summary of the dataset. The second one is the second best summary, independently from the first one, and so on. A small number of axes (minimum 2 to allow a 2D representation) has to be chosen by the user. Variables and individuals from the dataset are plotted on these axes, allowing the best representation(s) of the data table.

Several packages exist in R for PCA. Here we use the **ade4 package** which provides the **dudi.pca()** function ("dudi" stands for duality diagram, which is a theoretical concept for all multivariate ordination methods). The ade4 package must be installed.

```
dudi.pca(df, row.w = rep(1, nrow(df))/nrow(df),
    col.w = rep(1, ncol(df)), center = TRUE, scale = TRUE,
    scannf = TRUE, nf = 2)
```

`df` a data frame with n rows (individuals) and p columns (numeric variables)

`row.w` an optional row weights (by default, uniform row weights)

`col.w` an optional column weights (by default, unit column weights)

`center` a logical or numeric value, centring option
if TRUE, centring by the mean
if FALSE no centring
if a numeric vector, its length must be equal to the number of columns of the data frame df and gives the decentring

`scale` a logical value indicating whether the column vectors should be normed for the row.w weighting

`scannf` a logical value indicating whether the eigenvalue diagram should be displayed

`nf` if scannf FALSE, an integer indicating the number of kept axes

The following example uses the **meaudret** dataset (included in ade4 package), which comes from a small French river called the Meaudret. This is a list of 4 components:

env is a data frame with 20 sites and 9 variables (measurements).
spe is a data frame with 20 sites and 13 Ephemeroptera Species (Abundances).
design is a data frame with 20 sites and 2 factors.

- **season** is a factor with 4 levels-seasons.
- **site** is a factor with 5 levels-sites along the Meaudret river from S1 (upstream) to S5 (downstream).

spe.names is a character vector containing the scientific names of the 13 species.

NB: If you want to process your own data with R, you don't need to organize it as a list ! You can import separately each data.frame using the read.table() function.

```
# Chargement de la librairie ade4
library(ade4)

# ACP normée   ----

# ACP normée sur méaudret milieu ----
# dudi.pca() : exécuter l'ACP
# Diagramme des valeurs propres - screplot() et add.scatter.eig()
# s.corcircle() : cercle de corrélations des variables
# s.label() : nuage de points des individus

data(meaudret)
$env
     Temp Flow  pH Cond Bdo5 Oxyd  Ammo Nitr Phos
sp_1   10   41 8.5  295  2.3  1.4  0.12  3.4 0.11
sp_2   11  158 8.3  315  7.6  3.3  2.85  2.7 1.50
sp_3   11  198 8.5  290  3.3  1.5  0.40  4.0 0.10
sp_4   12  280 8.6  290  3.5  1.5  0.45  4.0 0.73
sp_5   13  322 8.5  285  3.6  1.6  0.48  4.6 0.84
su_1   13   62 8.3  325  2.3  1.8  0.11  3.0 0.13
su_2   13   80 7.6  380 21.0  5.7  9.80  0.8 3.65
su_3   15  100 7.8  385 15.0  2.5  7.90  7.7 4.50
su_4   16  140 8.0  360 12.0  2.6  4.90  8.4 3.45
su_5   15  160 8.4  345  1.7  1.9  0.22 10.0 1.74
au_1    1   25 8.4  315  1.6  0.5  0.07  6.4 0.03
au_2    3   63 8.0  425 36.0  8.0 12.50  2.2 6.50
au_3    2   79 8.1  350  7.1  1.9  2.70 13.2 3.70
au_4    3   85 8.3  330  2.0  1.4  0.42 12.0 1.60
au_5    2   72 8.6  305  1.6  0.9  0.10  9.5 1.25
wi_1    3  118 8.0  325  1.6  1.2  0.17  1.8 0.19
wi_2    3  252 8.3  360  9.5  2.9  2.52  4.6 1.60
wi_3    3  315 8.3  370  8.7  2.8  2.80  4.8 2.85
wi_4    3  498 8.3  330  4.8  1.6  1.04  4.4 0.82
wi_5    2  390 8.2  330  1.7  1.2  0.56  5.0 0.60

$design
        dat sta
sp_1 spring  S1
sp_2 spring  S2
sp_3 spring  S3
```

```
sp_4 spring  S4
sp_5 spring  S5
su_1 summer  S1
su_2 summer  S2
su_3 summer  S3
su_4 summer  S4
su_5 summer  S5
au_1 autumn  S1
au_2 autumn  S2
au_3 autumn  S3
au_4 autumn  S4
au_5 autumn  S5
wi_1 winter  S1
wi_2 winter  S2
wi_3 winter  S3
wi_4 winter  S4
wi_5 winter  S5

$spe
     Eda Bsp Brh Bni Bpu Cen Ecd Rhi Hla Hab Par Cae Eig
sp_1   4   7  10   9   0   0   0   5   9   0   4   0   0
sp_2   0   0   8   0   0   0   0   0   4   0   0   0   0
sp_3   0   5   5   0   0   0   0   2   5   0   0   0   0
sp_4   0   3   6   0   0   0   0   3   6   0   0   0   0
sp_5   0   5   6   0   0   0   5   0   4   0   0   0   4
su_1   6   7  10   0  10   0   0   2   7   0   0   0   2
su_2   0   0   9   0   0   0   0   0   0   0   0   0   0
su_3   0   6   8   0   0   2   0   0   0   0   0   0   0
su_4   0   7  11   0   0   2   0   0   2   0   0   5   5
su_5   0   6   9   2   3   0   4   0   0   0   0   2   7
au_1   4   5   8   0   9   6   0   5   9   0   7   0   0
au_2   0   0   1   0   0   0   0   0   0   0   0   0   0
au_3   0   9  10   0   0   0   0   0   4   0   3   0   0
au_4   0  10  13   0   0   3   0   5   5   1   4   2   4
au_5   2  10  12   0   4   0   8   4   4   2   5   1   6
wi_1   3   6   7   0   6   7   0   4   8   0   4   0   0
wi_2   0   3   6   0   0   5   0   4   3   0   1   0   0
wi_3   0   0   3   0   0   1   0   1   0   0   0   0   0
wi_4   0   6  10   0   0   5   1   3   5   0   2   0   0
wi_5   1   9  11   0   3   6   8   3   5   2   5   0   0

$spe.names
 [1] "Ephemera_danica"     "Baetis_sp"           "Baetis_rhodani"
 [4] "Baetis_niger"        "Baetis_muticus"      "Centroptilum_sp"
 [7] "Ecdyonorus_sp"       "Rhithrogena_sp"      "Habrophlebia_lauta"
[10] "Habroleptoides_sp"   "Paraleptophlebia_sp" "Caenis_sp"
[13] "Ephemerella_ignata"

is.list(meaudret)
[1] TRUE
```

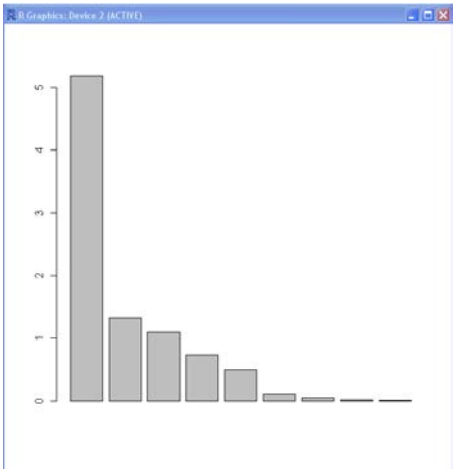
1.2. Normed PCA of the Meaudret environmental dataset

To analyze the environmental dataset, select the element **meaudret$env** as an object **mil**, of class data.frame:

```
mil <- meaudret$env
is.data.frame(mil)
[1] TRUE
```

Proceed with the PCA of **mil** data.frame, with **dudi.pca()** function, with defaults options (Normed PCA, because variables are in different units and must be standardized):

```
pca1 <- dudi.pca(mil)
```



First of all, the eigenvalue diagram is automatically plotted. Choose the number of axes you want to keep. Here the axis 1 is largely dominant, and axes 2 to 5 may be interesting also. Consequently we select 5 axes:

```
Select the number of axes: 5
```

If you want to avoid the printing of this diagram, you can use the **scannf=F** argument, and specify the number of axes to keep with **nf=** (here 5):

```
pca1 <- dudi.pca(mil, scannf=F, nf=5)

pca1
Duality diagramm
class: pca dudi
$call: dudi.pca(df = mil, scannf = F, nf = 5)

$nf: 5 axis-components saved
$rank: 9
eigen values: 5.175 1.32 1.093 0.7321 0.4902 ...
  vector length mode    content
1 $cw    9      numeric column weights
2 $lw    20     numeric row weights
3 $eig   9      numeric eigen values
```

```
  data.frame nrow ncol content
1 $tab       20   9    modified array
2 $li        20   5    row coordinates
3 $l1        20   5    row normed scores
4 $co        9    5    column coordinates
5 $c1        9    5    column normed scores
other elements: cent norm
```

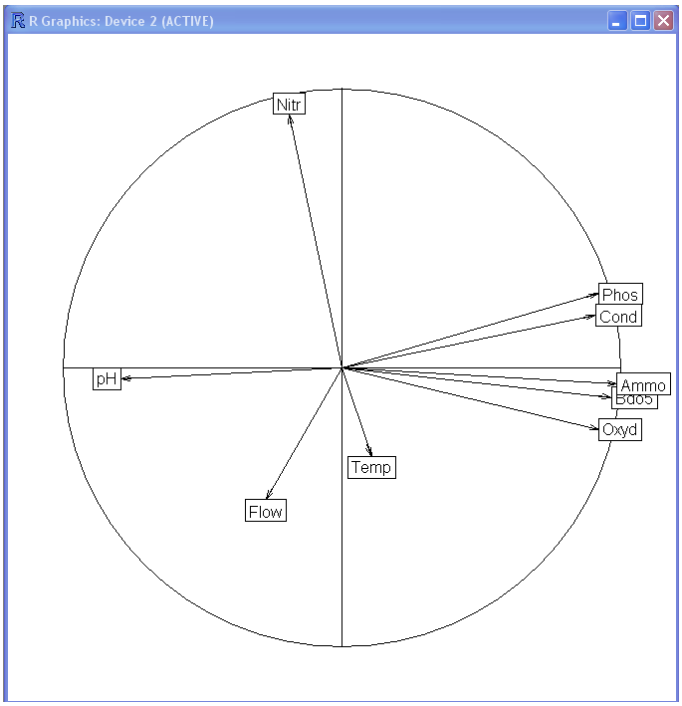
The percentage of inertia for each axis can be computed using **pca1$eig**:

```
pve <- 100 *pca1$eig / sum(pca1$eig)
print(pve, digits=2)
[1] 57.50 14.67 12.15  8.13  5.45  1.22  0.59  0.22  0.07
cpve <- cumsum(pve)
print(cpve, digits=3)
[1] 57.5  72.2  84.3  92.5  97.9  99.1  99.7  99.9 100.0
```

The first axis takes into account 57.5 % of the total inertia, the second one 14.7%. So the first factorial plane combining axes 1 and 2 represents 72.2%. This is a very good percentage. That means that 72% of the information from mil dataset can be summarized in only two dimensions.

In the case of Normed PCA, the variables (columns of the original data.frame) can be plotted on the axes as a correlation circle using the **s.corcircle()** function with the column coordinates in **pca1$co**. This file gives the coordinates for the 9 variables (in rows) on the 5 axes (in columns):

```
pca1$co
          Comp1       Comp2       Comp3       Comp4        Comp5
Temp  0.1054115 -0.32090155  0.83747545  0.42252564  0.057462741
Flow -0.2727582 -0.47432815 -0.58116930  0.60159191 -0.005507834
pH   -0.7934812 -0.04001985 -0.07599576 -0.07083901  0.589077994
Cond  0.9096049  0.18661194 -0.19409057  0.09858081 -0.114072990
Bdo5  0.9651860 -0.10767069 -0.06893578 -0.05723862  0.191404891
Oxyd  0.9208821 -0.22291370 -0.05260858 -0.10763822  0.229679553
Ammo  0.9846097 -0.05968657  0.02942241  0.01631680  0.041321363
Nitr -0.1898581  0.90635421  0.02252642  0.35220839  0.102644463
Phos  0.9203935  0.26401519 -0.04374686  0.19431145  0.158820642

s.corcircle(pca1$co,grid=F)
```



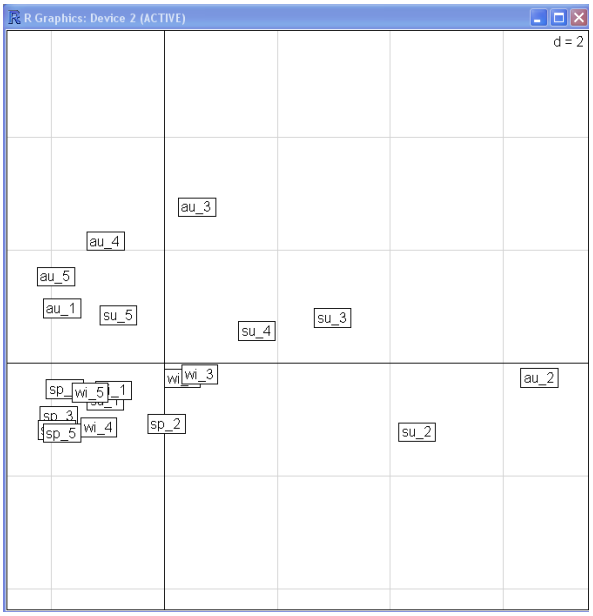
By default, the horizontal axis is axis 1 and the vertical one is axis 2. If you want to plot axes 3 vs. 4 you must use xax=3 and yax=4

```
s.corcircle(dfxy, xax = 1, yax = 2, label = row.names(df),
    clabel = 1, grid = TRUE, sub = "", csub = 1, possub = "bottomleft",
    cgrid = 0, fullcircle = TRUE, box = FALSE, add.plot = FALSE)
```

The individuals (rows of the original data.frame) can be plotted on the same axes using the **s.label()** function with the **pca1$li** coordinates (row coordinates):

```
pca1$li
           Axis1      Axis2      Axis3       Axis4       Axis5
sp_1 -1.75632208 -0.4635311  1.1291841 -1.12513814  0.34323496
sp_2  0.03943889 -1.0741475  0.6037606 -0.32789822  0.34618838
sp_3 -1.85986760 -0.9355197  0.6120348 -0.12167954  0.45328112
sp_4 -1.89974337 -1.1865046  0.3571121  0.47365766  0.88000303
sp_5 -1.80800144 -1.2412696  0.3737802  0.89404228  0.65295427
su_1 -1.03554353 -0.6609409  1.3689409 -0.64898974 -0.33952713
su_2  4.46933556 -1.2223981  0.9167693 -0.33951301 -0.97911738
su_3  2.97865187  0.8019368  1.1948545  1.14021714 -0.75265410
su_4  1.63544006  0.5710318  1.2685692  1.25659020 -0.21067502
su_5 -0.81194527  0.8493412  1.1138550  1.17906377  0.43937362
au_1 -1.80911039  0.9659537 -0.1617931 -1.51257831 -0.28708301
au_2  6.64098907 -0.2600388 -1.0674622 -1.06234887  1.38699616
au_3  0.57644364  2.7665446 -0.4558773  0.27231068 -0.13845161
au_4 -1.03856290  2.1633021 -0.2058648 -0.08993267 -0.01089342
au_5 -1.91924105  1.5347335 -0.2445503 -0.74513848  0.79710369
wi_1 -0.89575062 -0.4829240 -0.2640657 -1.24850350 -1.65642594
wi_2  0.32223564 -0.2709818 -1.2271403 -0.13673972  0.08919205
wi_3  0.63309494 -0.2004726 -1.5723832  0.44894654  0.17210601
wi_4 -1.15559971 -1.1350296 -2.0668474  1.12737242 -0.33056415
wi_5 -1.30594170 -0.5190854 -1.6728764  0.56625952 -0.85504156
```

```
s.label(pca1$li, boxes=T)
```



As for the s.corcircle function, by default, the horizontal axis is axis 1 and the vertical axis is axis 2. The boxes around the labels can be removed using **boxes=F** option. The 'd=2' in the upper right corner gives the scale of the graph: the value for d correspond to the size of a square side in the grid.

```
s.label(dfxy, xax = 1, yax = 2, label = row.names(dfxy),
    clabel = 1, pch = 20, cpoint = if (clabel == 0) 1 else 0, boxes = TRUE,
    neig = NULL, cneig = 2, xlim = NULL, ylim = NULL, grid = TRUE,
    addaxes = TRUE, cgrid = 1, include.origin = TRUE, origin = c(0,0),
    sub = "", csub = 1.25, possub = "bottomleft", pixmap = NULL,
    contour = NULL, area = NULL, add.plot = FALSE)
```

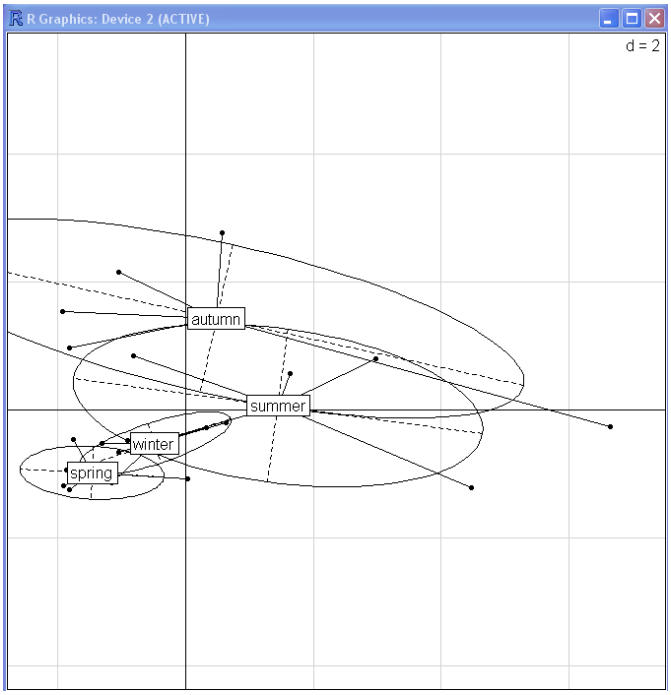
To interpret the result of this PCA, we must consider together the correlation circle and the plot of the individuals on the same axes.

Axis 1 is the most important one (57.5% of the total information). It opposes pH (left) to Phos, Cond, Ammo, Bdo5 and Oxyd (right). These 5 variables indicate a pollution occurring mostly in samples au_2, su_2, su_3, su_4. Axis 2 opposes Nitr (top) to Flow and Temp (bottom). Here, samples have been taken at 5 sites and 4 seasons, according to a factorial design, the **meaudret$design,** which can be used to group samples by site (sta) and season (dat) on the graph of the samples:

```
plan <- meaudret$design
plan
     season site
sp_1 spring  S1
sp_2 spring  S2
sp_3 spring  S3
sp_4 spring  S4
sp_5 spring  S5
```

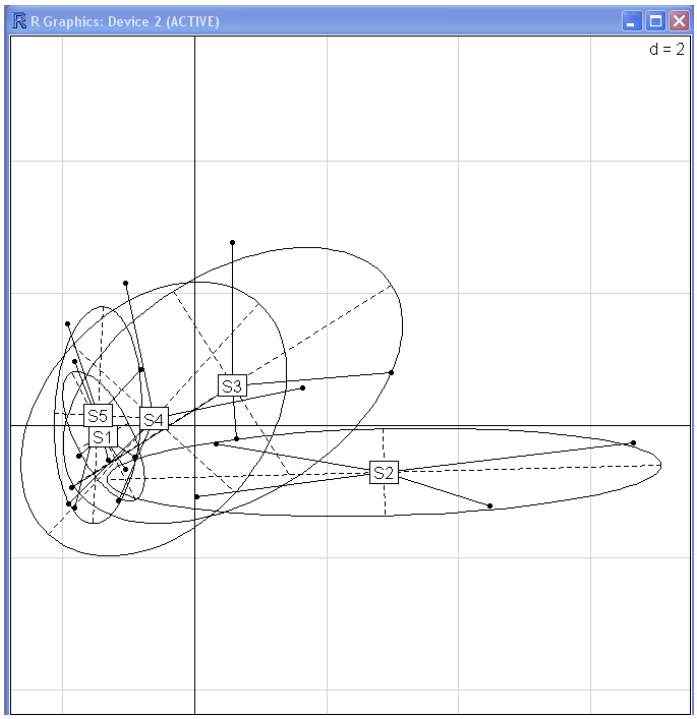
```
su_1 summer  S1
su_2 summer  S2
su_3 summer  S3
su_4 summer  S4
su_5 summer  S5
au_1 autumn  S1
au_2 autumn  S2
au_3 autumn  S3
au_4 autumn  S4
au_5 autumn  S5
wi_1 winter  S1
wi_2 winter  S2
wi_3 winter  S3
wi_4 winter  S4
wi_5 winter  S5
```

```
s.class(pca1$li,plan$season)
```



The samples belonging to the same season are grouped together by both a star and an ellipse, giving information about the average position of seasons on the factorial plan and of the heterogeneity of the group. Here, in autumn and summer, there is a high dispersion of samples (i.e. sites are very different from each other), and this dispersion mainly follows the first axis. On the contrary, in winter and spring, sites are very similar inducing a small dispersion.

```
s.class(pca1$li,plan$site)
```



The samples belonging to the same site are grouped together by both a star and an ellipse, giving information about the average position of sites on the factorial plan and of the heterogeneity of the group. For example here, site S2 shows a high seasonal variability along the first axis. S3 and S4 vary along the diagonal of axis 1 and 2. S1 and S5 are more homogeneous.

The interpretation is linked to a source of pollution occurring in summer just upstream S2 (touristic place). The pollution is measured by the variables Phos, Cond, Ammo, Bdo5 and Oxyd. In summer, sites S3 and S4 are also polluted, but they restore in autumn, whereas S2 remains polluted. S1 (upstream) is not polluted, neither is S5 because of the dilution in the river.

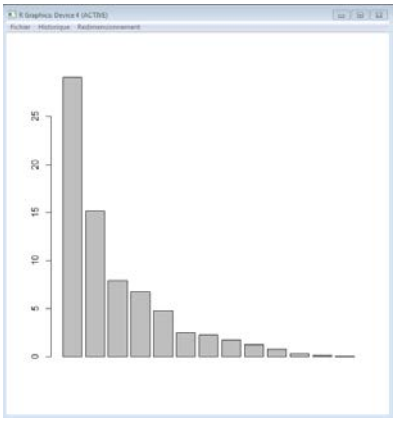
1.3. Centered PCA of the Meaudret faunistic dataset

Now let's try an application of **centered PCA,** using the same function **dudi.pca()** with the **scale=F** option. We use the faunistic dataset **spe** from meaudret. This dataset has 20 rows (the same as mil dataset) and 13 columns (13 Ephemeroptera species). On the contrary of mil dataset, all columns are in the same unit (abundance), that's why we are not obliged to standardize the data.

```
fau <- meaudret$spe
dim(fau)
[1] 20 13
head(fau)
     Eda Bsp Brh Bni Bpu Cen Ecd Rhi Hla Hab Par Cae Eig
sp_1   4   7  10   9   0   0   0   5   9   0   4   0   0
sp_2   0   0   8   0   0   0   0   0   4   0   0   0   0
sp_3   0   5   5   0   0   0   0   2   5   0   0   0   0
sp_4   0   3   6   0   0   0   0   3   6   0   0   0   0
```

```
sp_5   0   5   6   0   0   0   5   0   4   0   0   0   4
su_1   6   7  10   0  10   0   0   2   7   0   0   0   2
```

```
pca2 <- dudi.pca(fau , center=T, scale=F)
Select the number of axes: 2
```

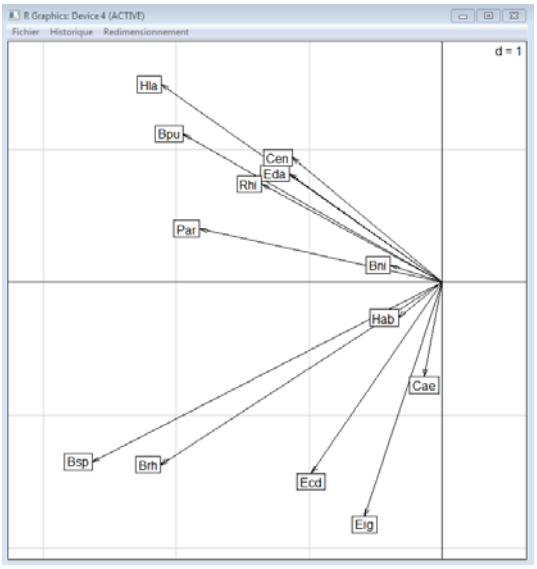


```
pve2 <- 100 *pca2$eig / sum(pca2$eig)
pve2
 [1] 40.09715840 20.94088479 10.90010998  9.31332737  6.60756834  3.42659291
 [7]  3.11120578  2.31256942  1.66618812  1.04428224  0.40961799  0.14344704
[13]  0.02704762

cumsum(pve2)
 [1]  40.09716  61.03804  71.93815  81.25148  87.85905  91.28564  94.39685
 [8]  96.70942  98.37561  99.41989  99.82951  99.97295 100.00000
```

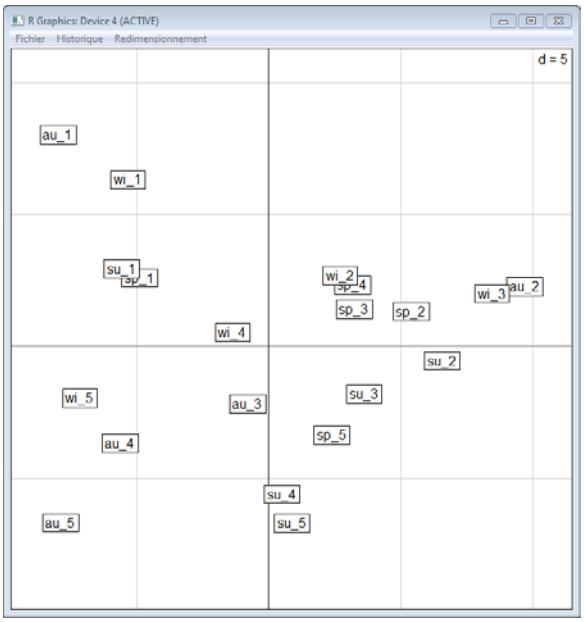
13 axes are computed. The first two axes represent 40.1% of the total inertia. **On the contrary of the Normed PCA, the columns cannot be plotted using a correlation circle and we must use the s.label() function for both rows and columns.** The **s.arrow()** function applied to the column coordinates gives the same graph as the s.label() with all the dots linked by arrows to the origin of axes. It is quite often used after a centred PCA by analogy with the correlation circle. Finally, the **s.class()** function can be used as well to group rows using the plan$season and plan$site factors.

```
s.label(pca2$co)
```

```
s.arrow(pca2$co)
```

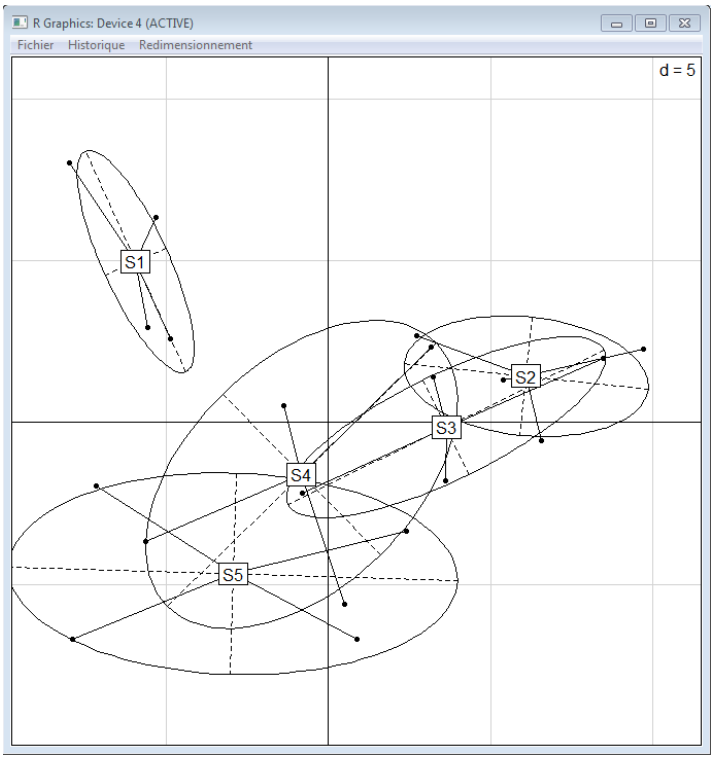



```
s.label(pca2$li)
```



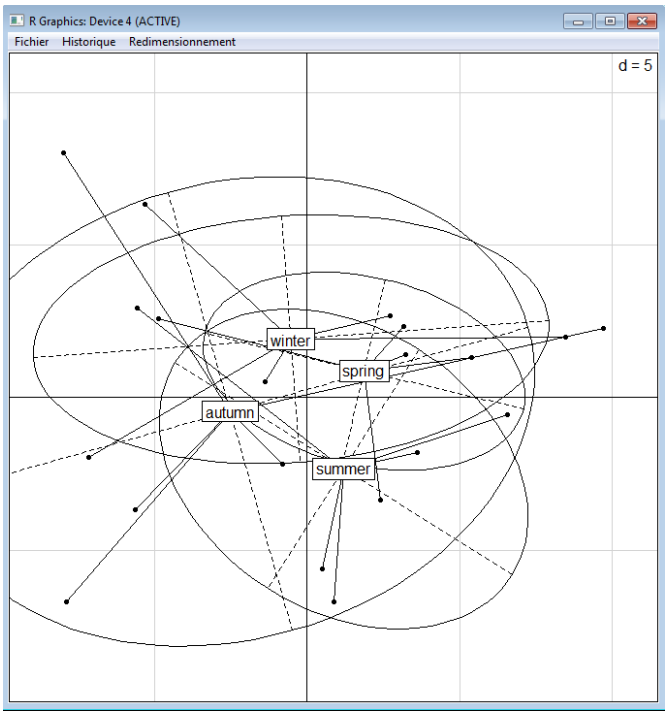
Notice that all species are projected on the left side of the first axis. This is called a “size effect”, opposing poor samples (on the right side: au_2, wi_3…) to rich samples (on the left side: au_1, sp_1, au_5, wi_5…). By checking the fau data.frame, we can confirm that au_2 and wi_3 contain only one or two Ephemeroptera. The axis 2 makes the difference between two groups of species. On the topleft side, the samples from site 1 present a particular community (Hla, Bpu, Par, Eda, Rhi, Cen). On the bottomleft, the other community (Bsp, Brh, Eig, Ecd…) corresponds to wi_5, au_5 and au_4. The s.class confirms this interpretation:

```
s.class(pca2$li, plan$site)
```



The samples belonging to the S1 site show a very stable community (Hla, Bpu…) whatever the season. S2 and S3 are also homogeneous, and they are characterized by their very poor community (maybe due to the pollution identified on the environmental PCA). Finally S4 and S5 are characterized by the second group of species (Bsp, Brh, Eig, Ecd…), but they can be very different according to the season inducing a large ellipse of dispersion.

```
s.class(pca2$li, plan$season)
```



The s.class plot according the season shows the high spatial heterogeneity of the Ephemeroptera community of the Meaudret river whatever the season.

2. Correspondence Analysis (COA or CA)

The **Correspondence Analysis** is used for the analysis of rectangular data table containing homogenous data. At the origin, COA was designed for the analysis of contingency tables, crossing two qualitative variables and giving in each cell the number of observations corresponding to both modalities. By extension, it is often used for sample x species abundance data. Here, in the meaudret example, it cannot be applied to the mil dataset because variables are in different units, but it is well suited to the fau dataset. The COA analysis proceeds to a double centring of the data table.

In the ade4 library, the correspondence analysis is done with the **dudi.coa()** function. It's syntax is very simple:

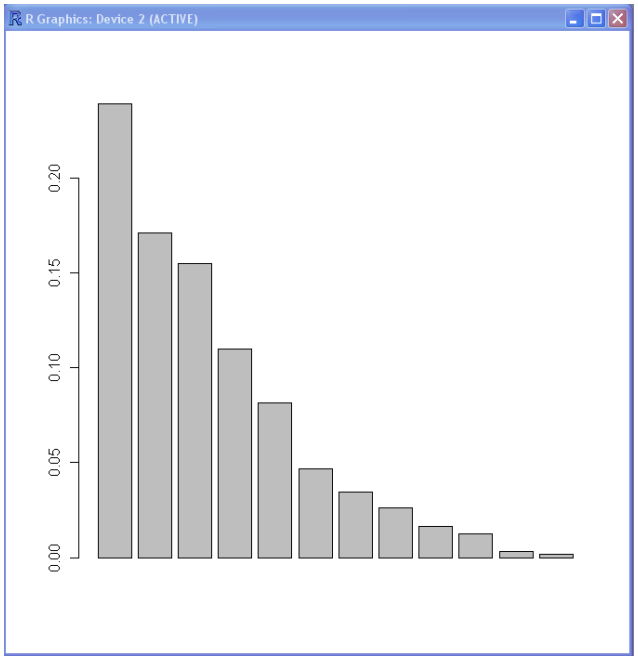
```
dudi.coa(df, scannf = TRUE, nf = 2)
```

`df` a data frame containing positive or null values

`scannf` a logical value indicating whether the eigenvalues bar plot should be displayed

`nf` if scannf FALSE, an integer indicating the number of kept axes

```
coa1 <- dudi.coa(fau)
Select the number of axes: 5
```



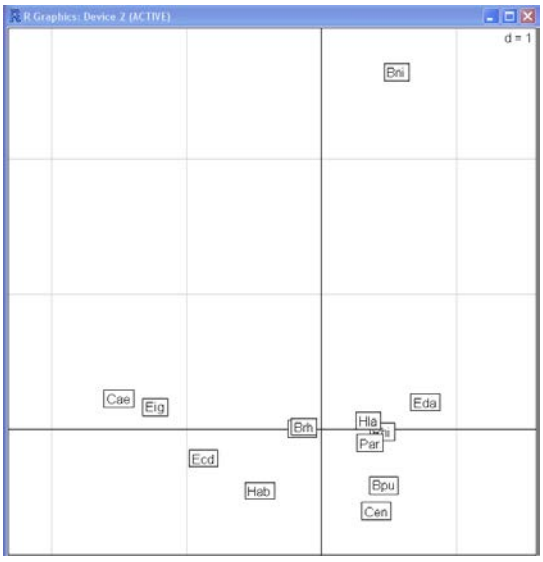
```
Duality diagramm
class: coa dudi
$call: dudi.coa(df = fau)

$nf: 5 axis-components saved
$rank: 12
eigen values: 0.239 0.1712 0.1547 0.1102 0.0814 ...
  vector length mode    content
1 $cw    13     numeric column weights
2 $lw    20     numeric row weights
3 $eig   12     numeric eigen values

  data.frame nrow ncol content
1 $tab       20   13   modified array
2 $li        20   5    row coordinates
3 $l1        20   5    row normed scores
4 $co        13   5    column coordinates
5 $c1        13   5    column normed scores
other elements: N
```

The eigenvalues diagram shows that 5 axes can be selected. Notice that p-1 axes are computed, where p is the smaller dimension of the table. It can be either rows or columns, because the COA is symmetrical.

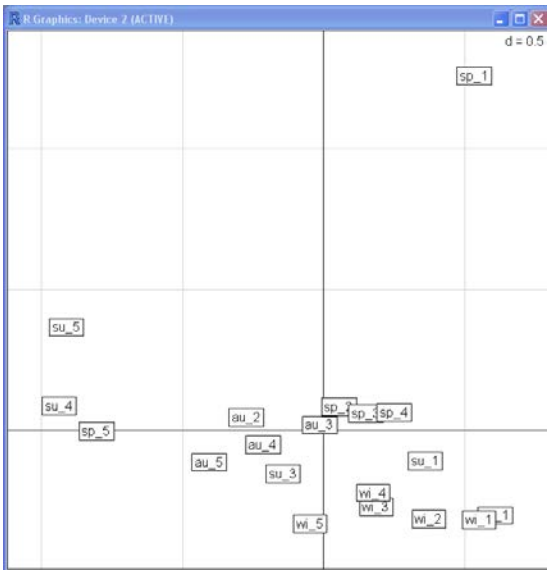
```
pve3 <- 100 *coa1$eig / sum(coa1$eig)
pve3
 [1] 26.6020217 19.0533583 17.2211442 12.2628243  9.0608274  5.2257712
 [7]  3.8329561  2.9279456  1.8409447  1.3991416  0.3653481  0.2077167
cumsum(pve3)
 [1]  26.60202  45.65538  62.87652  75.13935  84.20018  89.42595  93.25890
 [8]  96.18685  98.02779  99.42694  99.79228 100.00000
```

The first two axes represent 45.65% of the total inertia. Rows and columns are projected on the axes using the **s.label()** function as for the centered PCA analysis.

```
s.label(coa1$co)
```




```
s.label(coa1$li)
```



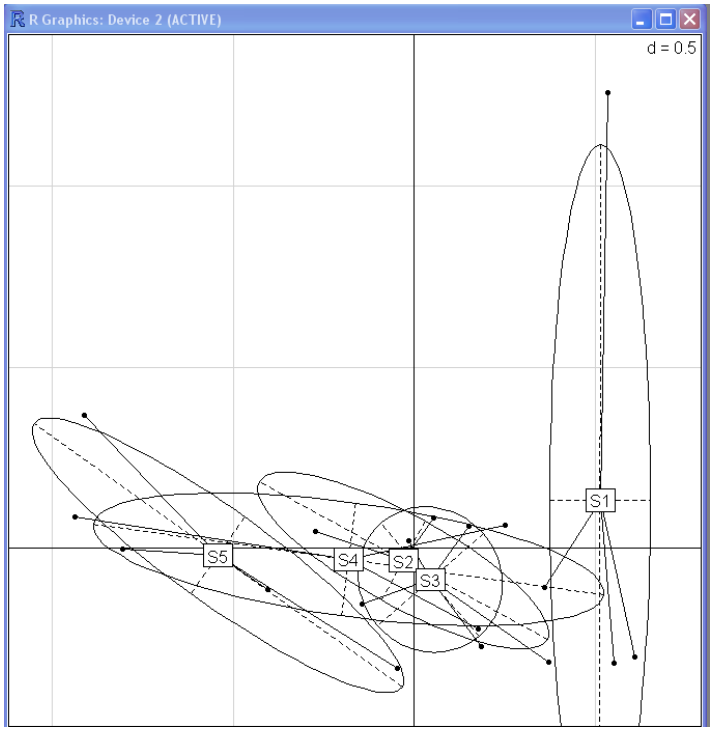
The Bni species, associated to the sp_1 sample is on the top of axis 2. We can check in the data table that Bni has been found only twice and with a high number in sample sp_1:

```
fau
     Eda Bsp Brh Bni Bpu Cen Ecd Rhi Hla Hab Par Cae Eig
sp_1   4   7  10   9   0   0   0   5   9   0   4   0   0
sp_2   0   0   8   0   0   0   0   0   4   0   0   0   0
sp_3   0   5   5   0   0   0   0   2   5   0   0   0   0
sp_4   0   3   6   0   0   0   0   3   6   0   0   0   0
sp_5   0   5   6   0   0   0   5   0   4   0   0   0   4
su_1   6   7  10   0  10   0   0   2   7   0   0   0   2
su_2   0   0   9   0   0   0   0   0   0   0   0   0   0
su_3   0   6   8   0   0   2   0   0   0   0   0   0   0
su_4   0   7  11   0   0   2   0   0   2   0   0   5   5
su_5   0   6   9   2   3   0   4   0   0   0   0   2   7
au_1   4   5   8   0   9   6   0   5   9   0   7   0   0
au_2   0   0   1   0   0   0   0   0   0   0   0   0   0
au_3   0   9  10   0   0   0   0   0   4   0   3   0   0
```

```
au_4   0  10  13   0   0   3   0   5   5   1   4   2   4
au_5   2  10  12   0   4   0   8   4   4   2   5   1   6
wi_1   3   6   7   0   6   7   0   4   8   0   4   0   0
wi_2   0   3   6   0   0   5   0   4   3   0   1   0   0
wi_3   0   0   3   0   0   1   0   1   0   0   0   0   0
wi_4   0   6  10   0   0   5   1   3   5   0   2   0   0
wi_5   1   9  11   0   3   6   8   3   5   2   5   0   0
```

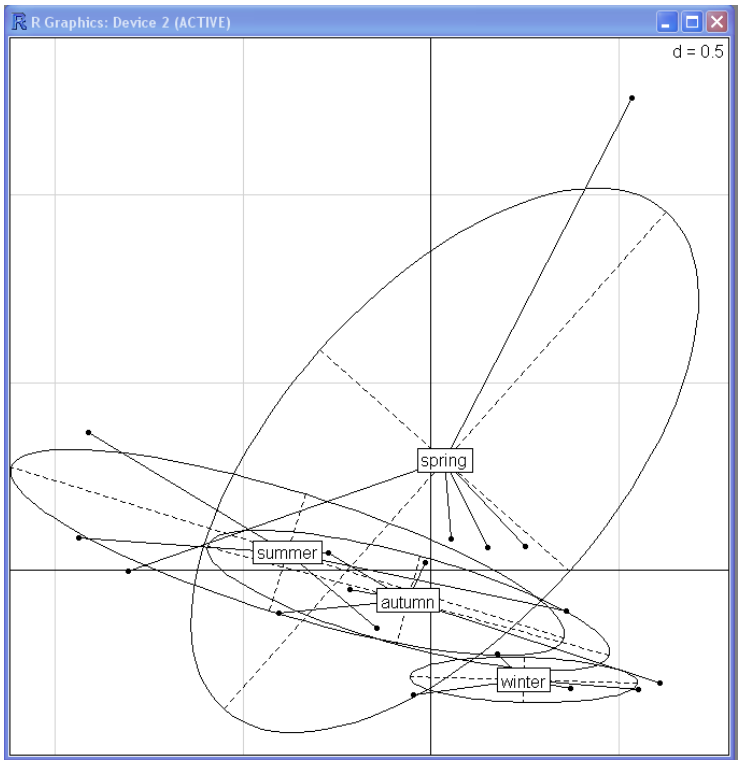
Let's now have a look to the s.class graphs to interpret the spatio-temporal structure:

```
s.class(coa1$li, plan$site)
```



The centers of the site classes follow an upstream-downstream gradient from right to left on the first axis. The S1 site however shows a high dispersion along the second axis, due to the sp_1 sample. The first axis can be considered as a spatial axis.

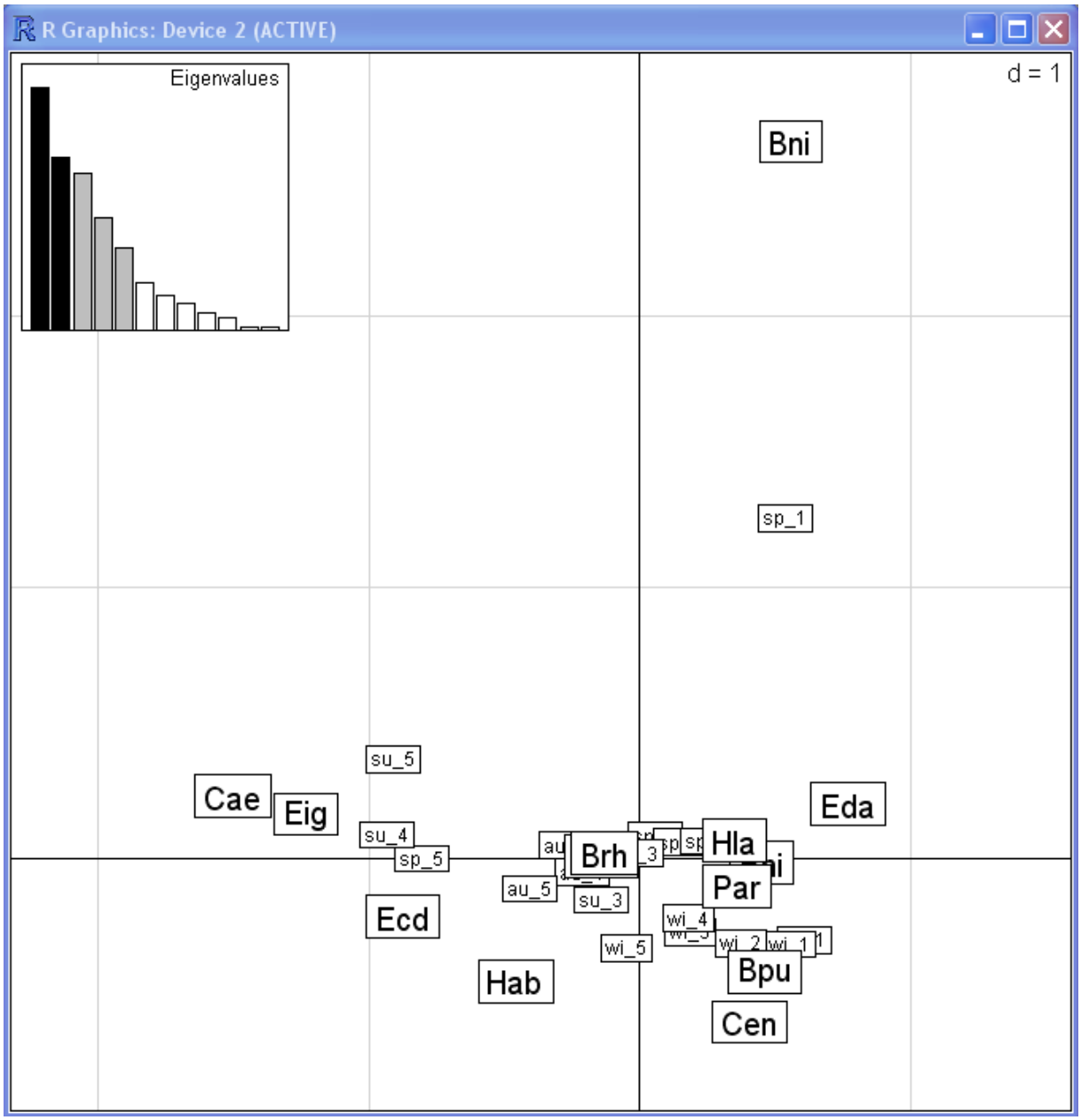
```
s.class(coa1$li, plan$season)
```



The centers of the season classes follow more or less a gradient from top to bottom on the axis 2 which can be considered as a seasonal axis. In spring, the dispersion is very high according to both axis 1 and 2. In summer, in autumn and to a lesser extent in winter, the dispersion of the sites follows mainly the axis 1.

The **scatter()** function produces the best graphical representation according to the class of the considered object (here dudi.coa). Here it produces a biplot graph with both rows and columns on the same graph and the eigenvalues diagram in the topleft corner:

```
scatter(coa1)
```

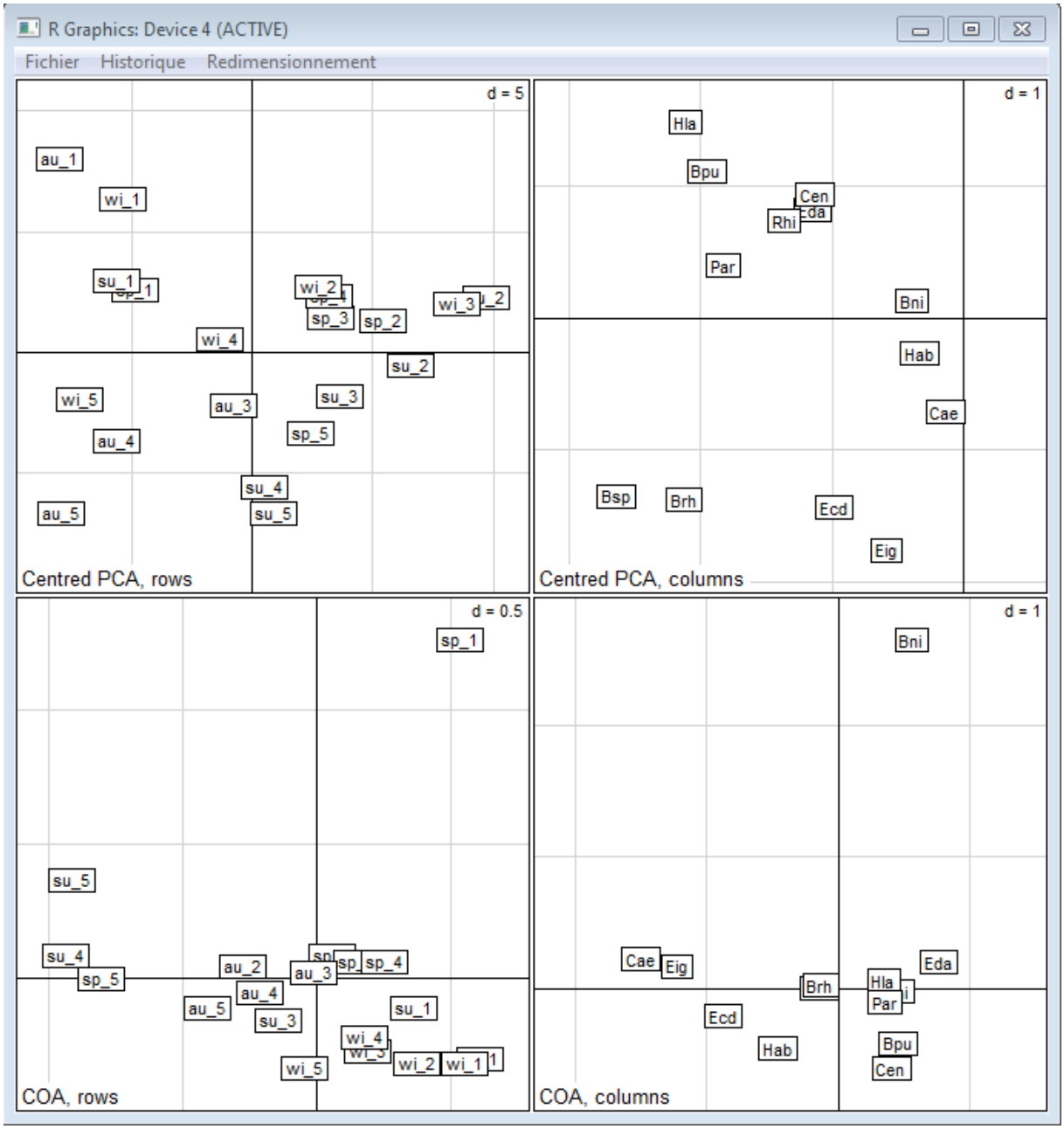


The conclusion of the Correspondence Analysis could be that the Ephemeroptera community in the Meaudret River follows an upstream-downstream gradient, except in spring due to the "Bni incident" (exceptional abundance of this species – try to rerun the COA without this species).

In this case, the COA is a good tool to understand the spatial and seasonal organization of the river… but where is the pollution effect previously identified on the PCA of this fau table??? In this PCA analysis, the S2 and S3 sites have been characterized as "poor" samples (remember the size effect on the first axis). In the COA, this doesn't appear. This is because **PCA deals with the absolute abundance of species**, and makes a difference between a sample with a lot of Ephemeroptera in it and a sample with very few Ephemeroptera. **The COA, on the contrary, deals with profiles** or proportions and two samples with the same faunistic profile will be considered as similar, whatever the total number of individuals in each. As the COA is symmetrical, it is the same for comparing

species, and that's why COA is known to give importance to rare species. Both methods are OK, depending on what you want to show. If you want to discuss about the whole community (including rare species), you'd better use COA. If you are mainly interested in dominant species, use the Centered PCA. The best is to try both of them!

```
par(mfrow=c(2,2))
s.label(pca2$li, sub="Centred PCA, rows") # upper left
s.label(pca2$co, sub="Centred PCA, columns") # upper right
s.label(coa1$li, sub="COA, rows") # lower left
s.label(coa1$co, sub="COA, columns") # lower right
```



The detail of the data transformation used by COA on a smaller dataset is given in the R script associated to this tutorial (see line 233-310)

3. Multiple Correspondence Analysis (MCA)

The Multiple Correspondence Analysis is a generalization of the Correspondence Analysis to more than two qualitative variables. It can be used to analyze a rectangular table where the columns are qualitative (factor) variables. The **dudi.acm()** function in the ade4 library deals with such a table. The qualitative variables must be coded as factors and the table is automatically transformed inside the dudi.acm function into a complete disjunctive table. A complete disjunctive table has as many columns as the total number of levels of all the factor variables and each column contains 0 or 1 according to the realization of the given modality for the statistical unit.

```
dudi.acm (df, row.w = rep(1, nrow(df)), scannf = TRUE, nf = 2)
```

`df` data frame containing only factors

`row.w` vector of row weights, by default, uniform weighting 1/n

`scannf` a logical value indicating whether the eigenvalues bar plot should be displayed (default=TRUE)

`nf` if scannf FALSE, an integer indicating the number of kept axes (default=2)

The syntax of the dudi.acm() function is very similar to the dudi.pca() and the dudi.coa() functions. The only compulsory information is df, the name of a dataframe with n rows (statistical units) and p columns (factor variables only). Row weights (in a numeric vector of length=n) can be given using row.w=. By defaults row weights are uniform (1/n). As usual scannf= and nf= can be used to deal with the axes selection.

Notice that the column weights cannot be chosen by user. They are automatically computed inside the function by the formula:

```
col.w <- apply(X, 2, function(x) sum(x * row.w))
```

where X is the complete disjunctive table.

In the following example, we will use the "**ours**" data frame from ade4 library.

The "ours" (bears) data frame has 38 rows, areas of the "Inventaire National Forestier", and 10 columns. This data frame contains the following columns:

1. altit: importance of the altitudinal area inhabited by bears, a factor with levels:
   - 1 less than 50% of the area between 800 and 2000 meters
   - 2 between 50 and 70%
   - 3 more than 70%
2. deniv: importance of the average variation in level by square of 50 km2, a factor with levels:
   - 1 less than 700m
   - 2 between 700 and 900 m
   - 3 more than 900 m
3. cloiso: partitioning of the massif, a factor with levels:
   - 1 a great valley or a ridge isolates at least a quarter of the massif
   - 2 less than a quarter of the massif is isolated

- 3 the massif has no split

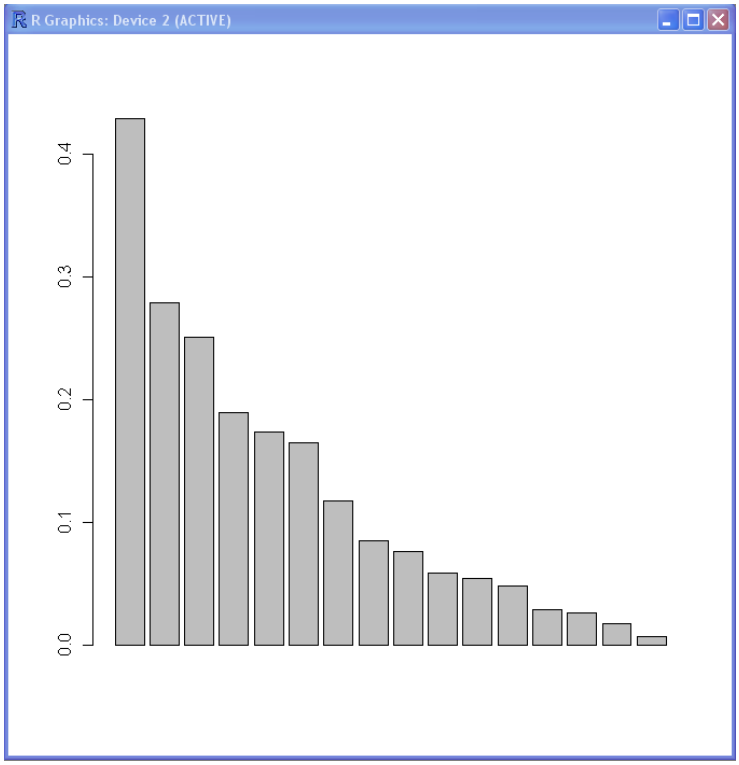
4. domain: importance of the national forests on contact with the massif, a factor with levels:
   - 1 less than 400 km2
   - 2 between 400 and 1000 km2
   - 3 more than 1000 km2
5. boise: rate of afforestation, a factor with levels:
   - 1 less than 30%
   - 2 between 30 and 50%
   - 3 more than 50%
6. hetra: importance of plantations and mixed forests, a factor with levels:
   - 1 less than 5%
   - 2 between 5 and 10%
   - 3 more than 10% of the massif
7. favor: importance of favorable forests, plantations, mixed forests, fir plantations, a factor with levels:
   - 1 less than 5%
   - 2 between 5 and 10%
   - 3 more than 10% of the massif
8. inexp: importance of unworked forests, a factor with levels:
   - 1 less than 4%
   - 2 between 4 and 8%
   - 3 more than 8% of the total area
9. citat: presence of the bear before its disappearance, a factor with levels:
   - 1 no quotation since 1840
   - 2 1 to 3 quotations before 1900 and none after
   - 3 4 quotations before 1900 and none after
   - 4 at least 4 quotations before 1900 and at least 1 quotation between 1900 and 1940
10. depart: district, a factor with levels:
    - AHP Alpes-de-Haute-Provence
    - AM Alpes-Maritimes
    - D Drôme
    - HP Hautes-Alpes
    - HS Haute-Savoie
    - I Isère
    - S Savoie


```
data(ours)
summary(ours)
```

```
 altit  deniv  cloiso domain boise  hetra  favor  inexp  citat  depart
 1: 8   1:13   1:12   1: 9   1:10   1:19   1:15   1:20   1:22   AHP:5
 2:17   2:14   2: 4   2:13   2:15   2: 5   2:12   2:10   2: 7   AM :4
 3:13   3:11   3:22   3:16   3:13   3:14   3:11   3: 8   3: 4   D  :5
                                                      4: 5   HP :8
                                                             HS :4
                                                             I  :5
                                                             S  :7
```

Variables 1 to 8 provide a qualitative description of the habitat, variables 9 and 10 are illustrative variables. We use only variables 1 to 8 to run a Multiple Correspondence Analysis with the dudi.acm() function:

```
acm <- dudi.acm(ours[, 1:8])
```



We decide to keep 2 axes, but at least 3 axes could be considered.

```
acm
Duality diagramm
class: acm dudi
$call: dudi.acm(df = ours[, 1:8], scannf = FALSE)

$nf: 2 axis-components saved
$rank: 16
eigen values: 0.4283 0.2783 0.2501 0.1888 0.1733 ...
  vector length mode    content
1 $cw    24     numeric column weights
2 $lw    38     numeric row weights
3 $eig   16     numeric eigen values
```

```
  data.frame nrow ncol content
1 $tab       38   24   modified array
2 $li        38   2    row coordinates
3 $l1        38   2    row normed scores
4 $co        24   2    column coordinates
5 $c1        24   2    column normed scores
other elements: cr
```

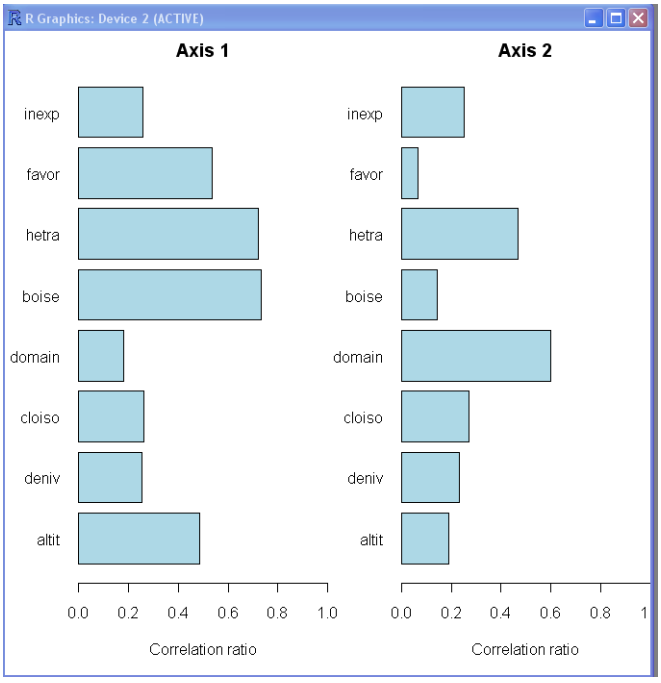
The coordinates of the 38 rows onto the selected axes are in the file **acm$li**. The coordinates of the 24 modalities of the 8 variables are in the file **acm$co**.

A new element is computed for MCA: **cr**, the correlation ratio between the variables and the axes:

```
acm$cr
             RS1       RS2
altit  0.4867906 0.1893131
deniv  0.2528392 0.2325247
cloiso 0.2598395 0.2724303
domain 0.1796741 0.6002266
boise  0.7317972 0.1429825
hetra  0.7233694 0.4699515
favor  0.5352790 0.0682467
inexp  0.2568897 0.2505504
```

Let's plot this correlation ratio as a barplot for axis 1 and axis 2:

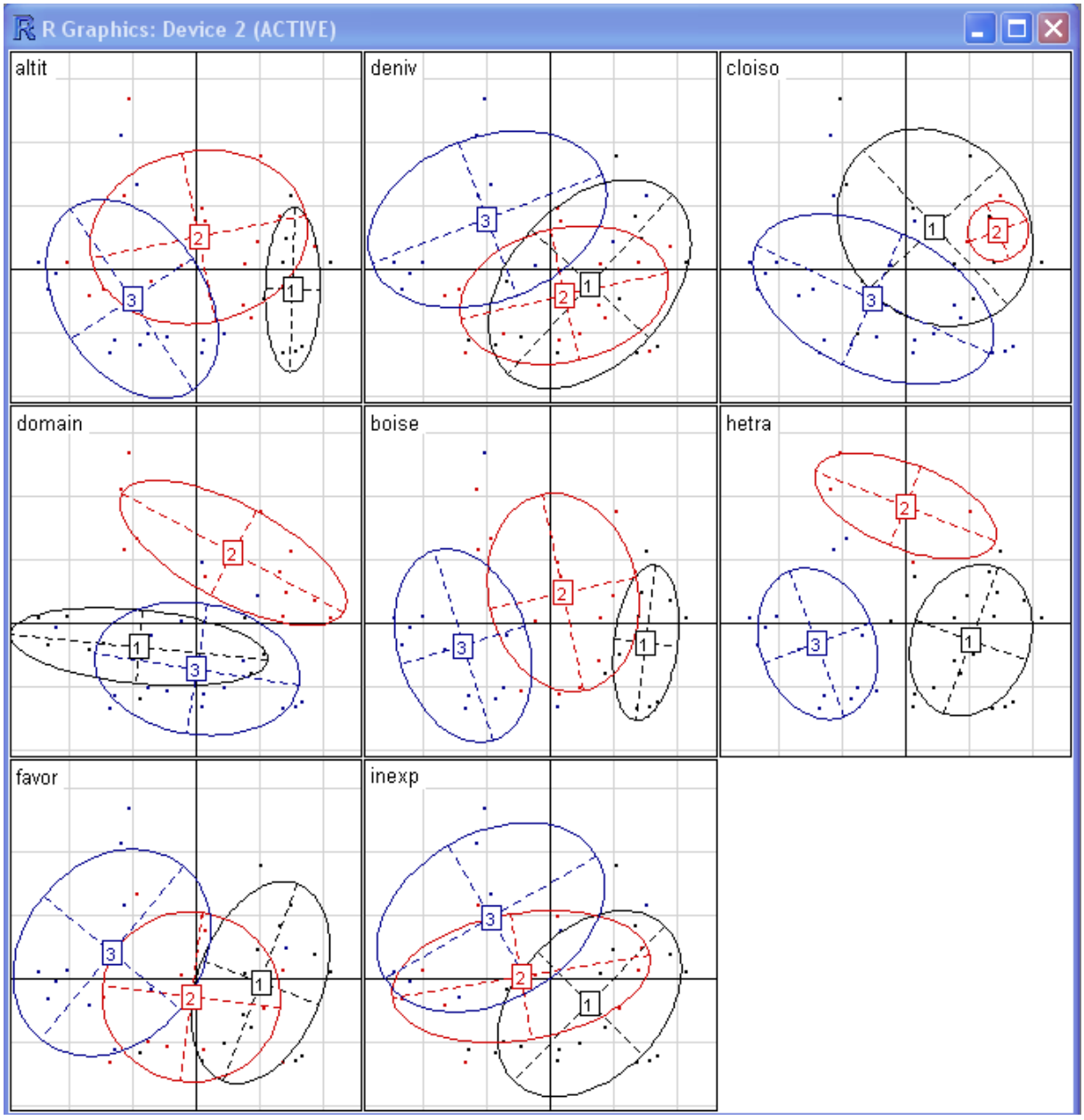
```
par(mfrow = c(1, 2), mar = c(5, 4, 2, 0))
barplot(acm$cr[, 1], horiz = TRUE, xlim = c(0, 1), names.arg =
colnames(ours[1:8]),las = 1, main = "Axis 1", col = "lightblue", xlab =
"Correlation ratio")
barplot(acm$cr[, 2], horiz = TRUE, xlim = c(0, 1), names.arg =
colnames(ours[1:8]),las = 1, main = "Axis 2", col = "lightblue", xlab = "
Correlation ratio ")
```



Variables hetra, boise, favor and altit are well correlated with axis 1 (high cr values), while domain and hetra are well correlated with axis 2.

The **scatter()** function produces the best graphical representation according to the class of the considered object (here dudi.acm). The 8 qualitative variables are projected (like with the s.class function) together with the 38 points corresponding to the rows of the data set. Here, as usual, axis 1 is horizontal and axis 2 is vertical. For the "altit" variable for example, the first axis opposes modality 1 to modality 3. The 'domain' variable, with a high correlation ratio to axis 2 shows on this axis an opposition between modality 2 (positive values) to modalities 1 and 3 (negative values).

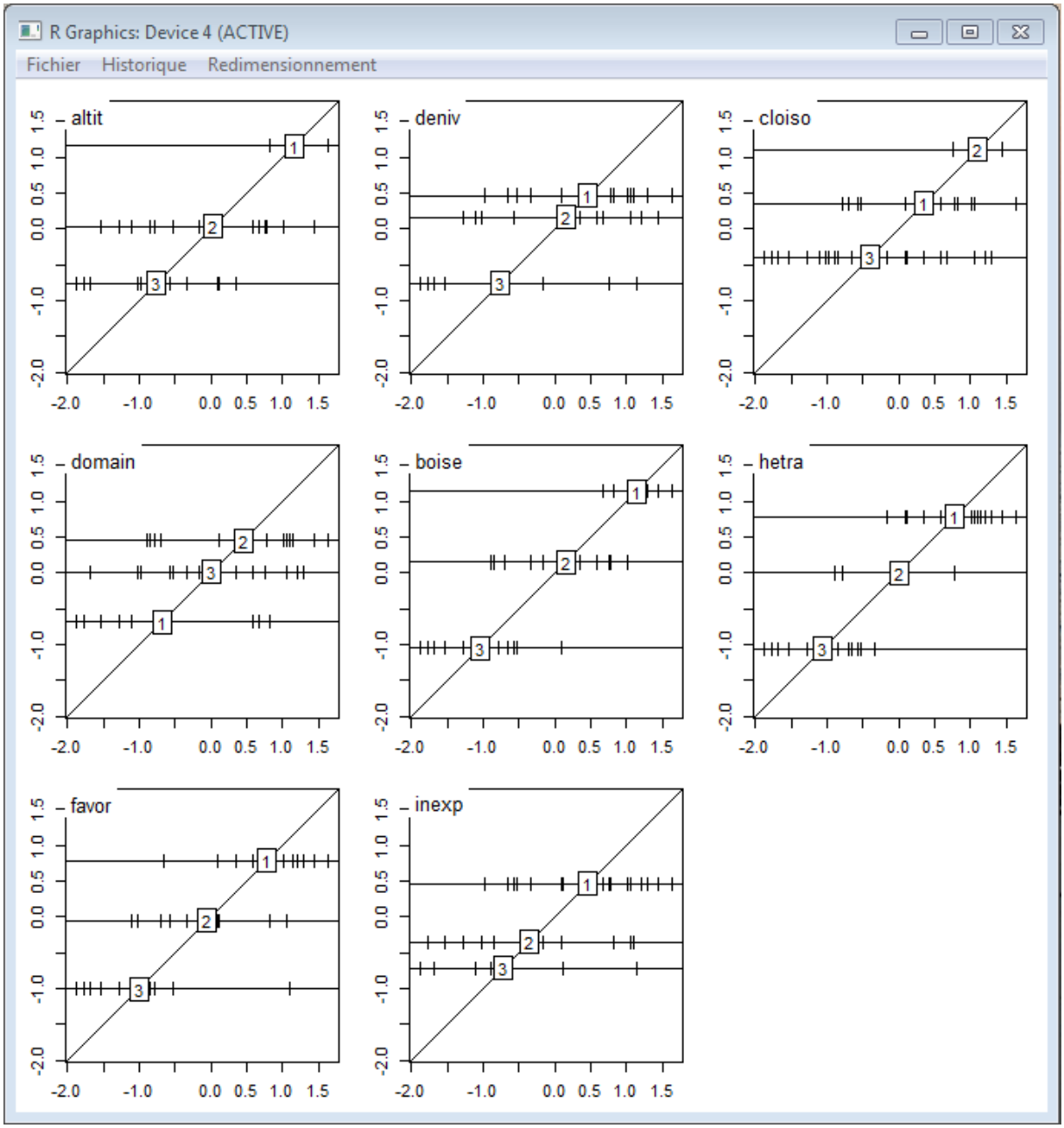
```
scatter(acm, col = rep(c("black", "red3", "darkblue"), 2))
```



Another useful help to interpret the results of an MCA is given by the **score()** function which is a univariate version of the 2D graph above. It can be particularly useful in the case when

the axis 1 is clearly higher than the others. For each variable, the rows are ordered horizontally by their score on the considered axis and vertically by the score of the modality they belong to. The score of a modality is the average score of the rows belonging to this modality.

```
score(acm, xax = 1)
```



4. Automatic Classification Methods

Automatic Classification Methods aim at classifying the elements of a data set (the statistical units or individuals) into groups, *i.e.* to create a **partition** of this dataset. The imposed constraints are that each group must be **as homogeneous as possible**, and that groups must be **as different as possible from each other**. One can search for a hierarchy of groups, which is a **binary tree** also called **dendrogram**, or for a partition of individuals, without hierarchy.

4.1. Distances between objects

The search for a hierarchy between individuals, or **hierarchical clustering** uses the concept of distances between individuals. This concept implies a measure of **heterogeneity** within a group (based on the distances between individuals belonging to it) and a measure of **dissimilarity** between two groups (based on the distance between an individual from one group and an individual from the other group).

The **dist()** function allows to compute a matrix of distance between individuals which are the rows of a data matrix of quantitative data (or in 0/1). One can specify different computation methods. The result of the function is an R object of class dist.

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE)
```

By default, euclidean distances are computed (well suited to continuous numeric variables). Other possibilities can be specified by the **method=** argument. The **diag=** and **upper=** arguments specify respectively if the diagonal and the upper triangle of the distance matrix should be printed. By default they are not.

There are numerous methods for computing distances between individuals. In the ade4 library, there are specific functions dedicated to the computation of distances for different kinds of data. The **dist.binary()** function is dedicated to ecological distances and allows to choose among 10 different distances. The **dist.quant()** function allows to compute some distances on quantitative data such as Mahalanobis distance, generally used for morphometric data. The **dist.genet()** function computes any one of five measures of genetic distance from a set of gene frequencies in different populations with several loci.

4.2. Hierarchical clustering: the hclust() function

The **hclust()** function performs a **hierarchical cluster analysis** using a set of dissimilarities (or distances) for the n individuals being clustered. Initially, each individual is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster. At each stage, distances between clusters are recomputed by the Lance–Williams dissimilarity update formula according to the particular clustering method being used.

A number of different clustering methods are provided. ***Ward's*** minimum variance method aims at finding compact, spherical clusters. The ***complete linkage*** method finds similar

clusters. The ***single linkage*** method (which is closely related to the minimal spanning tree) adopts a 'friends of friends' clustering strategy. The other methods (for example ***average***) can be regarded as aiming for clusters with characteristics somewhere between the single and complete link methods.

```
hclust(d, method = "complete")
```

d is the name of the distance matrix to be processed. The default method is "complete".

An object of class **hclust** describes the tree produced by the clustering process. The tree can be plotted using the **plot()** function (specific method for class hclust).

```
plot(h, hang=-1)
```

where h is an object of class hclust. The argument **hang=-1** allows to get all the labels on the same line.

The tree can be "cut" into n groups by using the **cutree()** function :

```
cutree(h, k=n)
```

where h is an object of class hclust. The argument **k=n** allows to cut the tree in order to get n groups of individuals. The number of groups should be chosen by the user after visualization of the tree, in order to cut the longest branches of the tree.

Example of using hclust and cutting the resulting tree to get a partition of the 150 individuals from iris data.

First we compute the distance matrix using only the first 4 columns (because the 5th one is not numerical):

```
iris.dist <- dist(iris[,1:4], method="euclidean")
iris.dist
           1         2         3         4         5         6         7
2   0.5385165
3   0.5099020 0.3000000
4   0.6480741 0.3316625 0.2449490
5   0.1414214 0.6082763 0.5099020 0.6480741
6   0.6164414 1.0908712 1.0862780 1.1661904 0.6164414
7   0.5196152 0.5099020 0.2645751 0.3316625 0.4582576 0.9949874
...
```

If there is a large number of individuals, you should better avoid printing the distance matrix which is nxn.

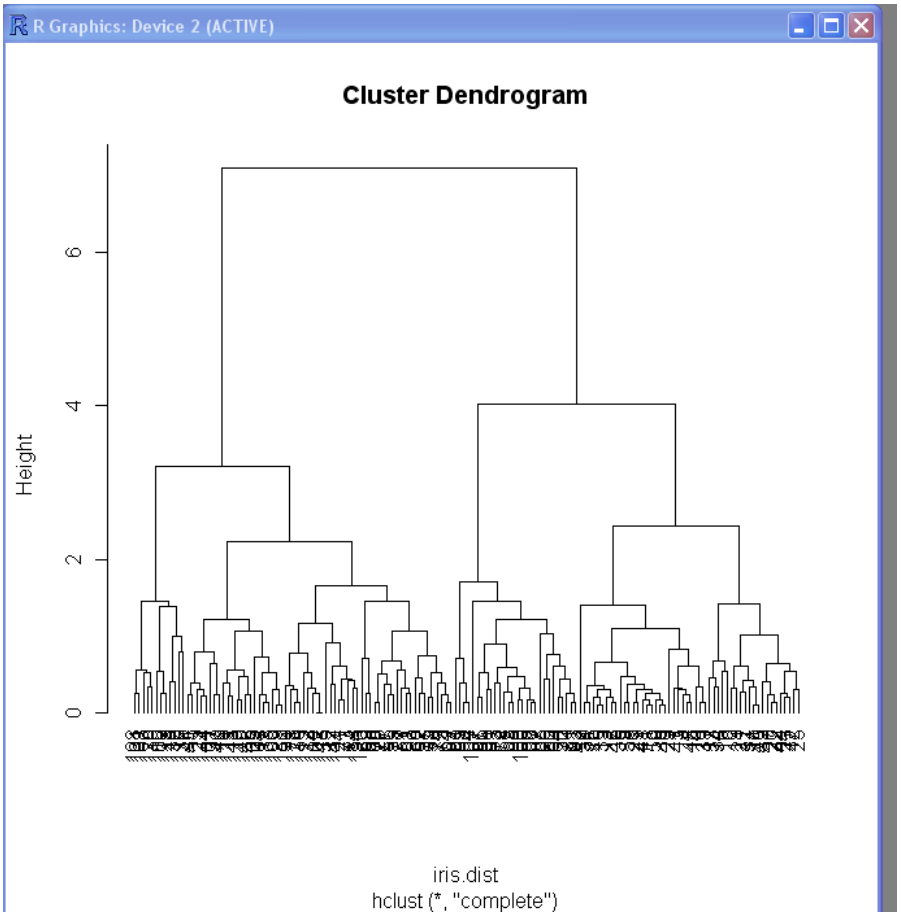
Then compute the hierarchy using the default method:

```
iris.hclust.lc <- hclust(iris.dist)  # default method : complete link
iris.hclust.lc

Call:
hclust(d = iris.dist)
```

```
Cluster method   : complete
Distance         : euclidean
Number of objects: 150
```

Plot the tree (dendrogram). Try to use or not the `hang=-1` argument to see the difference.

```
plot(iris.hclust.lc, hang=-1)
```

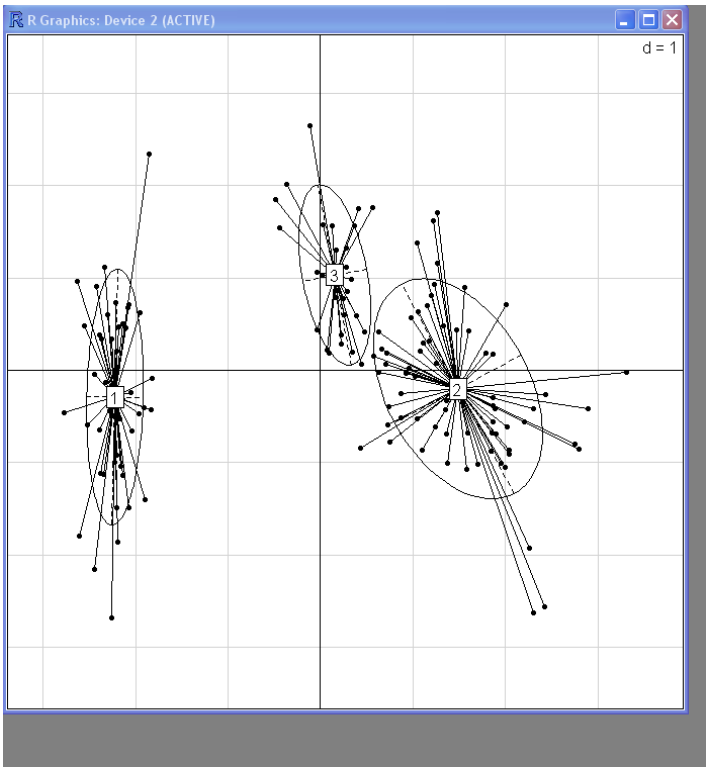


It is impossible here to read the labels, due to the high number of individuals. A dendrogram can be used only for a small set of individuals (less than ~50). You can then cut the tree, choosing (for example) 2, 3 or 4 groups. Here we choose k=3. The result is a vector of the same length as the number of individuals (150) giving for each individual the number of the group he has been assigned to:

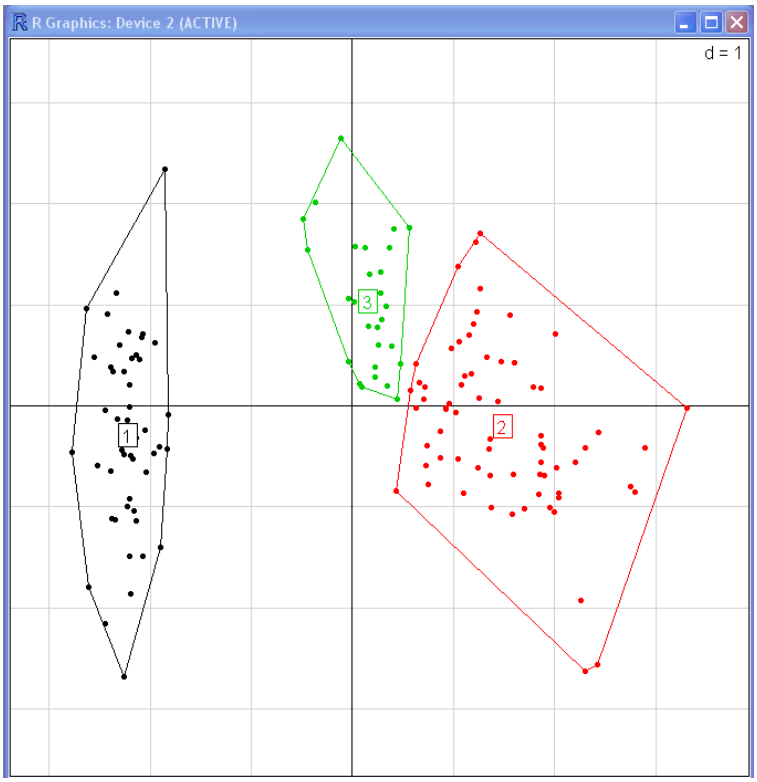
```
iris.k.lc <- cutree(iris.hclust.lc, k=3)
iris.k.lc
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 3 2 3 2 3 3 3 3 2 3 2 3 3 2 3 2 3 2 2
 [75] 2 2 2 2 2 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 2 3 3 2 2 2 2 2 2 3 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2
```

You can use this number to project groups on the PCA axes using s.class() or s.chull() functions:

```
iris.pca <- dudi.pca(iris[,1:4], scannf=F, nf=2)

s.class(iris.pca$li, as.factor(iris.k.lc))
```

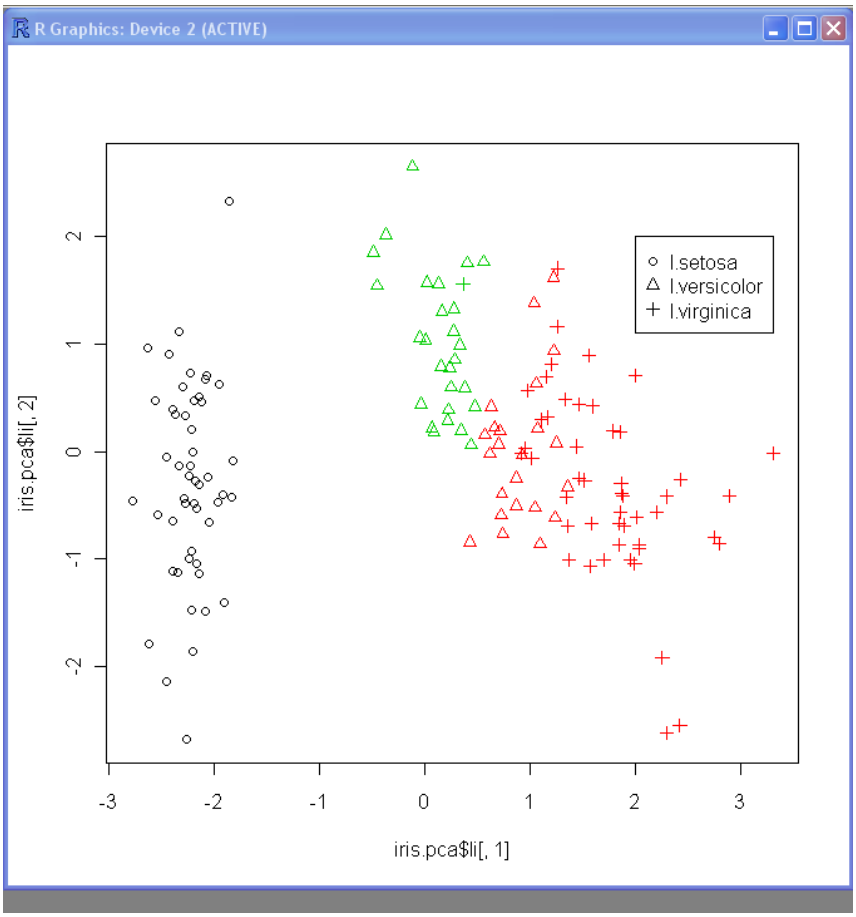



```
s.chull(iris.pca$li, as.factor(iris.k.lc), optchull =1,
cpoint=1,col=c(1,2,3))
```



You can also compare it (in this particular case) to the existing partition of individuals by the Species variable on the PCA axes. Here we use the plot() function because it allows different colors (col=) and symbols (pch=):

```
plot(iris.pca$li[,1],iris.pca$li[,2],
     col=iris.k.lc,
     pch=(1:3)[iris$Species])
legend(2, 2, c("I.setosa", "I.versicolor", "I.virginica"),pch=1:3)
```

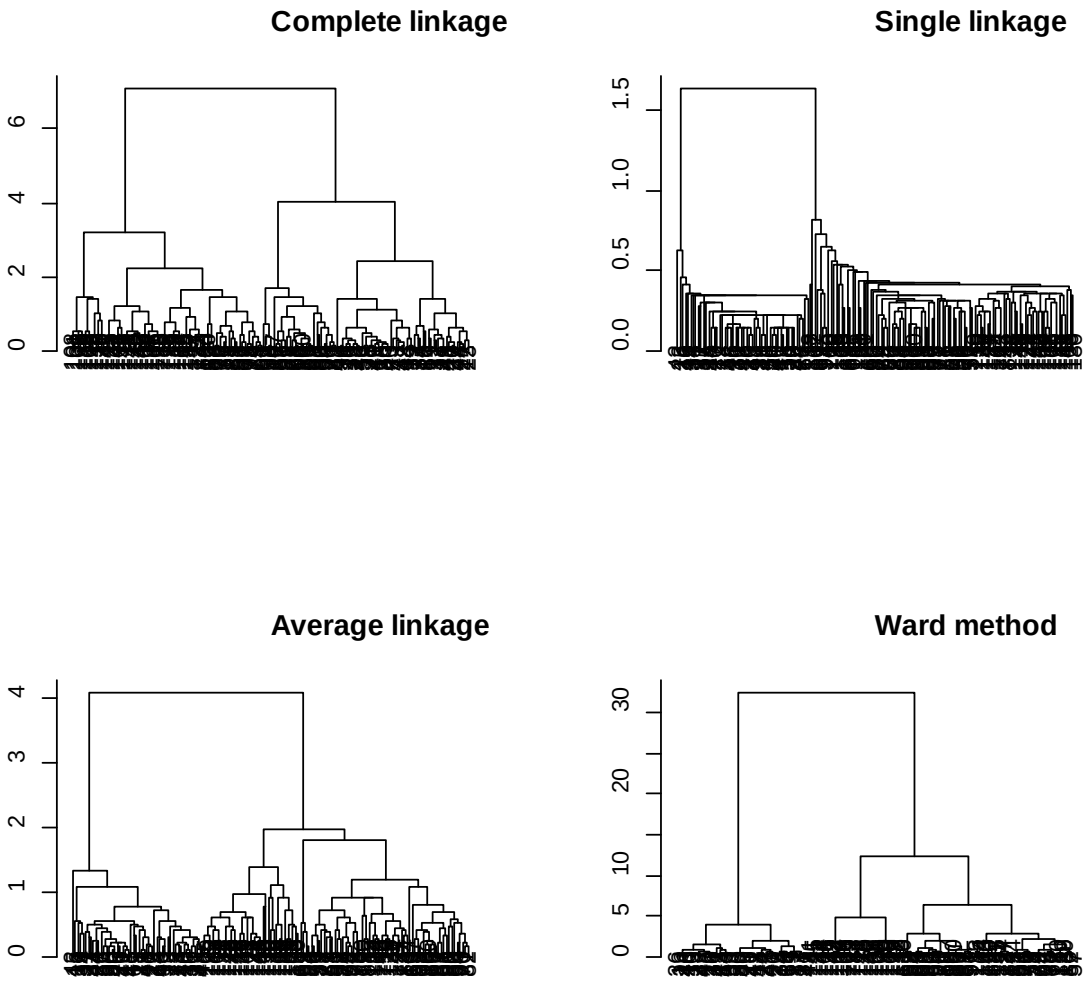


We can conclude that the setosa Species is really different from the others, considering the 4 numeric variable measured on the sample, whereas some I. versicolor are associated by the automatic clustering to the same group as I. virginica (red triangles).

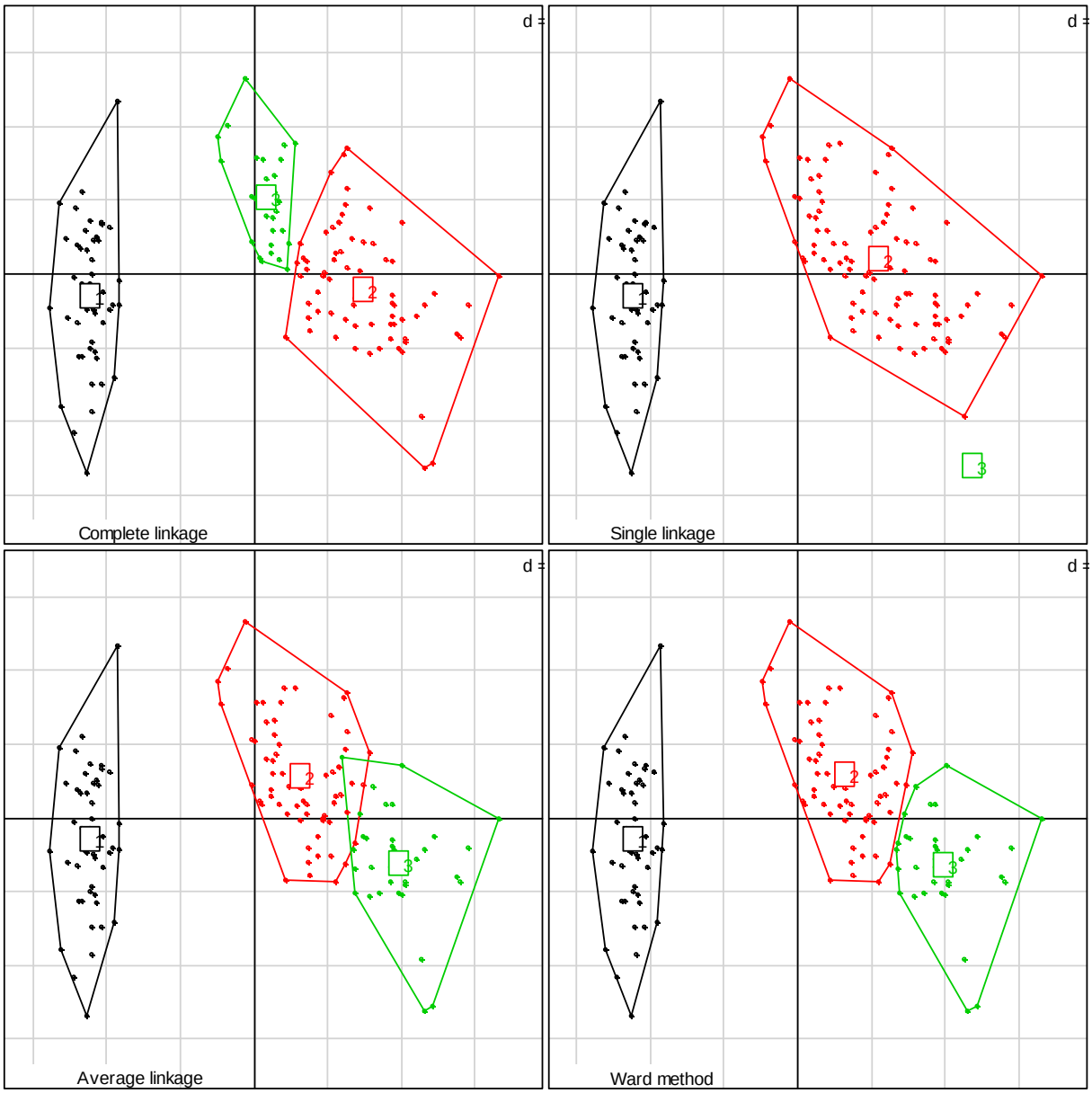
The choice on the aggregation method can lead to very different dendrograms and partitions:

```
# Single linkage
iris.hclust.ls <- hclust(iris.dist, method = "single")
iris.hclust.ls
iris.k.ls <- cutree(iris.hclust.ls, k=3)
table(iris$Species, iris.k.ls)
# Average linkage
iris.hclust.lm <- hclust(iris.dist, method = "average")
iris.hclust.lm
```

```
iris.k.lm <- cutree(iris.hclust.lm, k=3)
table(iris$Species, iris.k.lm)
# Ward
iris.hclust.wa <- hclust(iris.dist, method = "ward.D2")
iris.hclust.wa
iris.k.wa <- cutree(iris.hclust.wa, k=3)
table(iris$Species, iris.k.wa)

par(mfrow=c(2,2))
plot(iris.hclust.lc, hang=-1, main="Complete linkage", xlab="", ylab="",
sub="")
plot(iris.hclust.ls, hang=-1, main="Single linkage", xlab="", ylab="",
sub="")
plot(iris.hclust.lm, hang=-1, main="Average linkage", xlab="", ylab="",
sub="")
plot(iris.hclust.wa, hang=-1, main="Ward method", xlab="", ylab="", sub="")
```



```
par(mfrow=c(2,2))
s.chull(iris.pca$li, as.factor(iris.k.lc), optchull =1,
cpoint=1,col=c(1,2,3), sub="Complete linkage")
s.chull(iris.pca$li, as.factor(iris.k.ls), optchull =1,
cpoint=1,col=c(1,2,3), sub="Single linkage")
s.chull(iris.pca$li, as.factor(iris.k.lm), optchull =1,
cpoint=1,col=c(1,2,3), sub="Average linkage")
s.chull(iris.pca$li, as.factor(iris.k.wa), optchull =1,
cpoint=1,col=c(1,2,3), sub="Ward method")
```



4.3 Looking for a partition: the kmeans() function

The **kmeans()** function can be used if you want to get a partition of your data, without building a hierarchy. If the number of individuals is very large, the kmeans method can be the only one available because the computation of a distance matrix for a very large number of

individuals will require too much memory and it will be impossible to use the distance-based hierarchical method.

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))
```

`x`	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
`centers`	either the number of clusters, say *k*, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in `x` is chosen as the initial centres.
`iter.max`	the maximum number of iterations allowed.
`nstart`	if `centers` is a number, how many random sets should be chosen?
`algorithm`	character: may be abbreviated. Note that `"Lloyd"` and `"Forgy"` are alternative names for one algorithm.
`object`	an **R** object of class `"kmeans"`, typically the result `ob` of `ob <- kmeans(..)`.
`method`	character: may be abbreviated. `"centers"` causes `fitted` to return cluster centers (one for each input point) and `"classes"` causes `fitted` to return a vector of class assignments.
`trace`	logical or integer number, currently only used in the default method (`"Hartigan-Wong"`): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.

The data given by x is clustered by the **k-means** method, which aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centers is minimized.

Different methods can be used. The algorithm of Hartigan and Wong (1979) is used by default. Note that some authors use k-means to refer to a specific algorithm rather than the general method: most commonly the algorithm given by MacQueen (1967) but sometimes that given by Lloyd (1957) and Forgy (1965). The Hartigan–Wong algorithm generally does a better job than either of those, but trying several random starts (nstart>1) is often recommended. For ease of programmatic exploration, k=1 is allowed, notably returning the center and withinss.

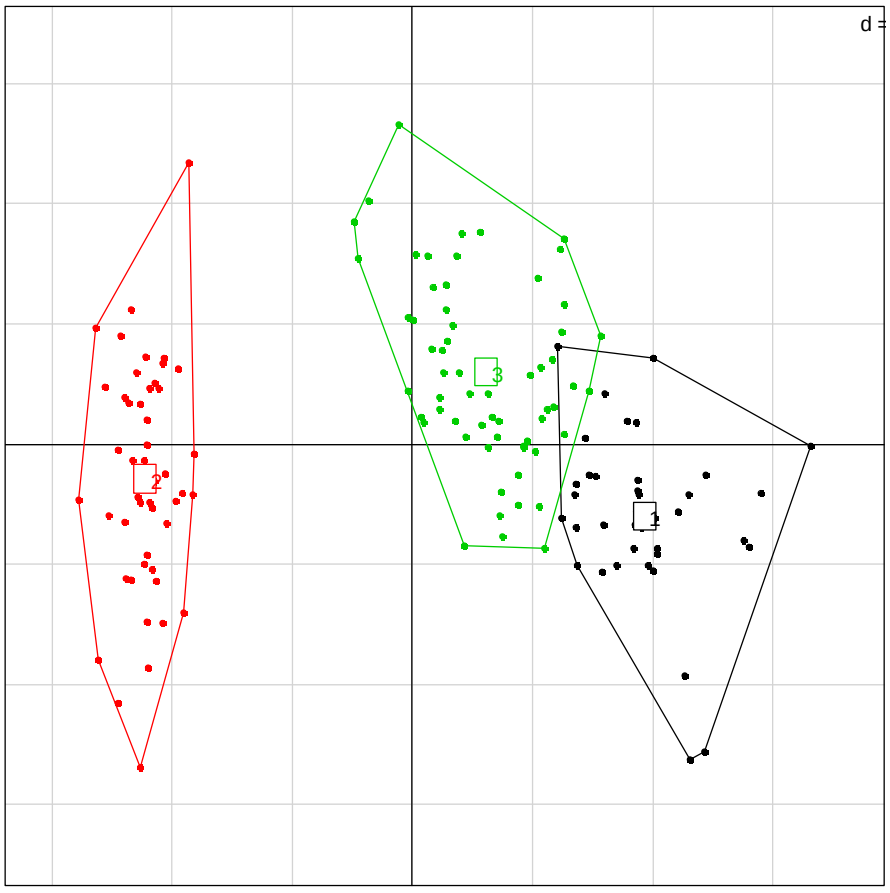
Except for the Lloyd–Forgy method, k clusters will always be returned if a number is specified. If an initial matrix of centres is supplied, it is possible that no point will be closest to one or more centres, which is currently an error for the Hartigan–Wong method.

First we apply this method to the iris dataset to compute a partition of the 150 individuals into 3 groups.

```
cl.iris <- kmeans(iris[,1:4], 3)
cl.iris
K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
```

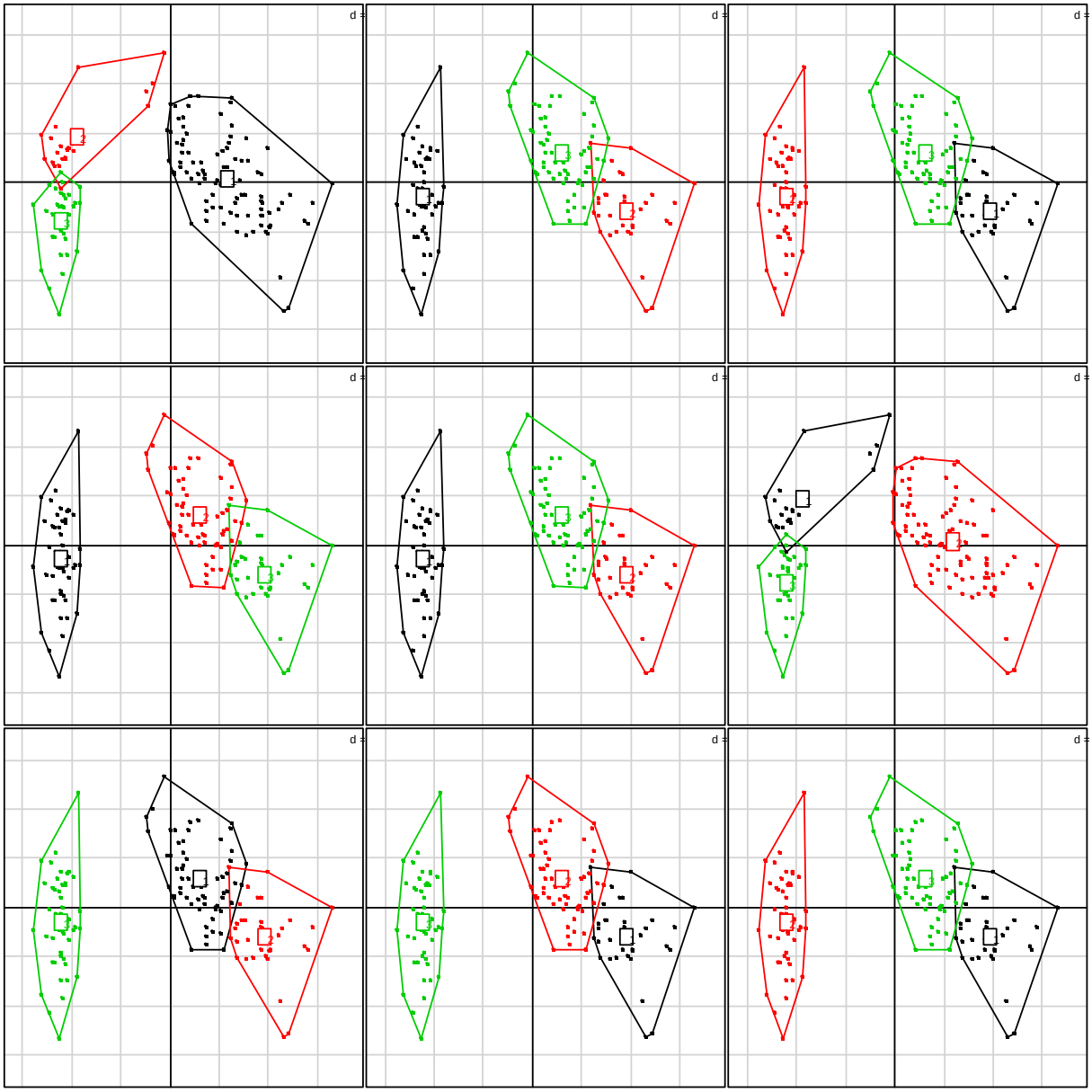
```
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
 [52] 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3
[103] 1 1 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

```
s.chull(iris.pca$li,      as.factor(cl.iris$cluster),      optchull      =1,
cpoint=1,col=c(1,2,3))
```



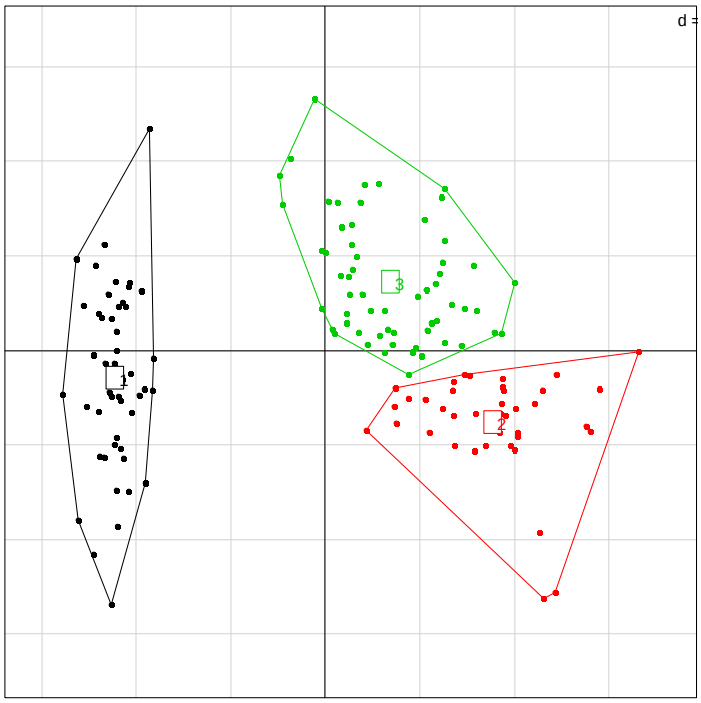
If we repeat several times the kmeans() procedure, we obtain very different partitions, as the result is highly influenced by the random initial partition.

```
par(mfrow=c(3,3))
for (i in 1:9) {
     cl.iris <- kmeans(iris[,1:4], 3)
     s.chull(iris.pca$li, as.factor(cl.iris$cluster), optchull =1,
cpoint=1,col=c(1,2,3))
}
```

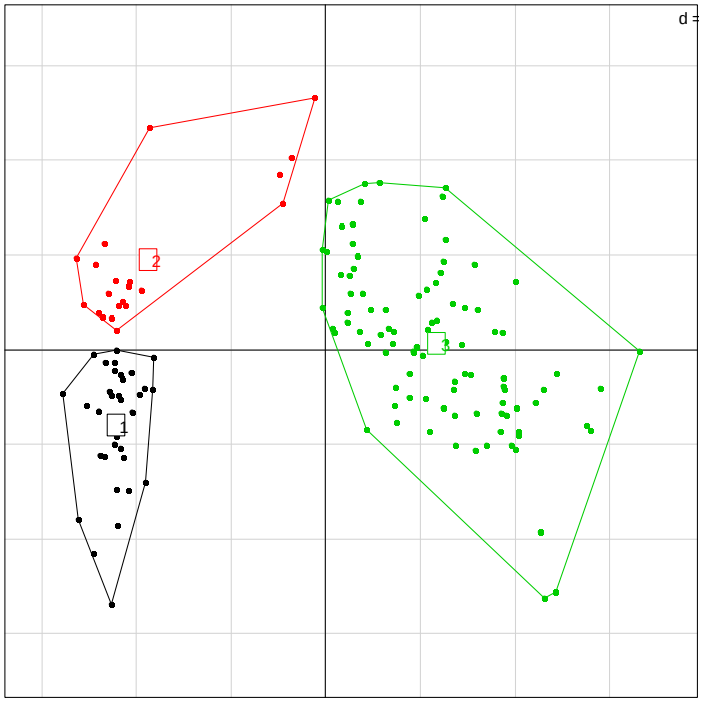


Instead of the raw data, the CAH and the kmean can be applied on the row coordinates from a dudi analysis (PCA, COA, MCA…).

```
# CAH
iris.dist <- dist(iris.pca$li, method="euclidean")
iris.hclust.acp <- hclust(iris.dist, method="ward.D2")
iris.hclust.acp
plot(iris.hclust.acp, hang=-1)
iris.k.acp <- cutree(iris.hclust.acp, k=3)
iris.k.acp
# Projection de ces groupes sur l'ACP du tableau iris
s.chull(iris.pca$li, as.factor(iris.k.acp), optchull =1, cpoint=1,
col=c(1,2,3))
```




```
# Kmeans
cl.iris.acp <- kmeans(iris.pca$li, 3)
s.chull(iris.pca$li, as.factor(cl.iris.acp$cluster), optchull =1,
cpoint=1,col=c(1,2,3))
```



See also the very complete document (in French): Introduction Classification Hierarchique_Chessel_etal.pdf

Bibliography

PDF files

Champely S. (2005) Introduction à l analyse multivariée (factorielle).pdf

Cours ade4, Novembre 2010, Université de Lyon :

- **Partie2-1_Initiation_ACP.pdf**
- **Partie2-2_ACP.pdf**
- **Partie2-3_Initiation_AFC.pdf**
- **Partie2-4_Initiation_ACM.pdf**
- **Partie2_5_Ordination_Tableaux_Ecologiques.pdf**

Introduction Classification Hierarchique_Chessel_etal(fr).pdf : Fiche de Biostatistique Stage 7, par D. Chessel, J. Thioulouse et A.B. Dufour.

tdr69_classification.pdf par A.B. Dufour et S. Dray

Ade4 WebSite

http://pbil.univ-lyon1.fr/ADE-4 : all the pdf files can be downloaded from this website