Table of Contents

Section: Ordination analysis

Ordination diagrams

Theory R functions **Examples** Exercise

R is notoriously not good in drawing ordination diagrams with complex information, because these usually need manual adjustment, which is not easy in R. Generally, if you need ordination diagram with interpretation focused on individual species or samples (with appropriate labels), R may not produce satisfying results - you may consider using CANOCO 4.5 with CanoDraw for Windows or CANOCO 5[1)]. However, R can still draw nice ordination diagrams, if the focus is on the overall pattern of samples, the grouping of samples into clusters or projecting linear or surface response of environment onto ordination space (even three dimensional); few examples are below.

The following plotting functions are using results of DCA (detrended correspondence analysis) of Vltava data:

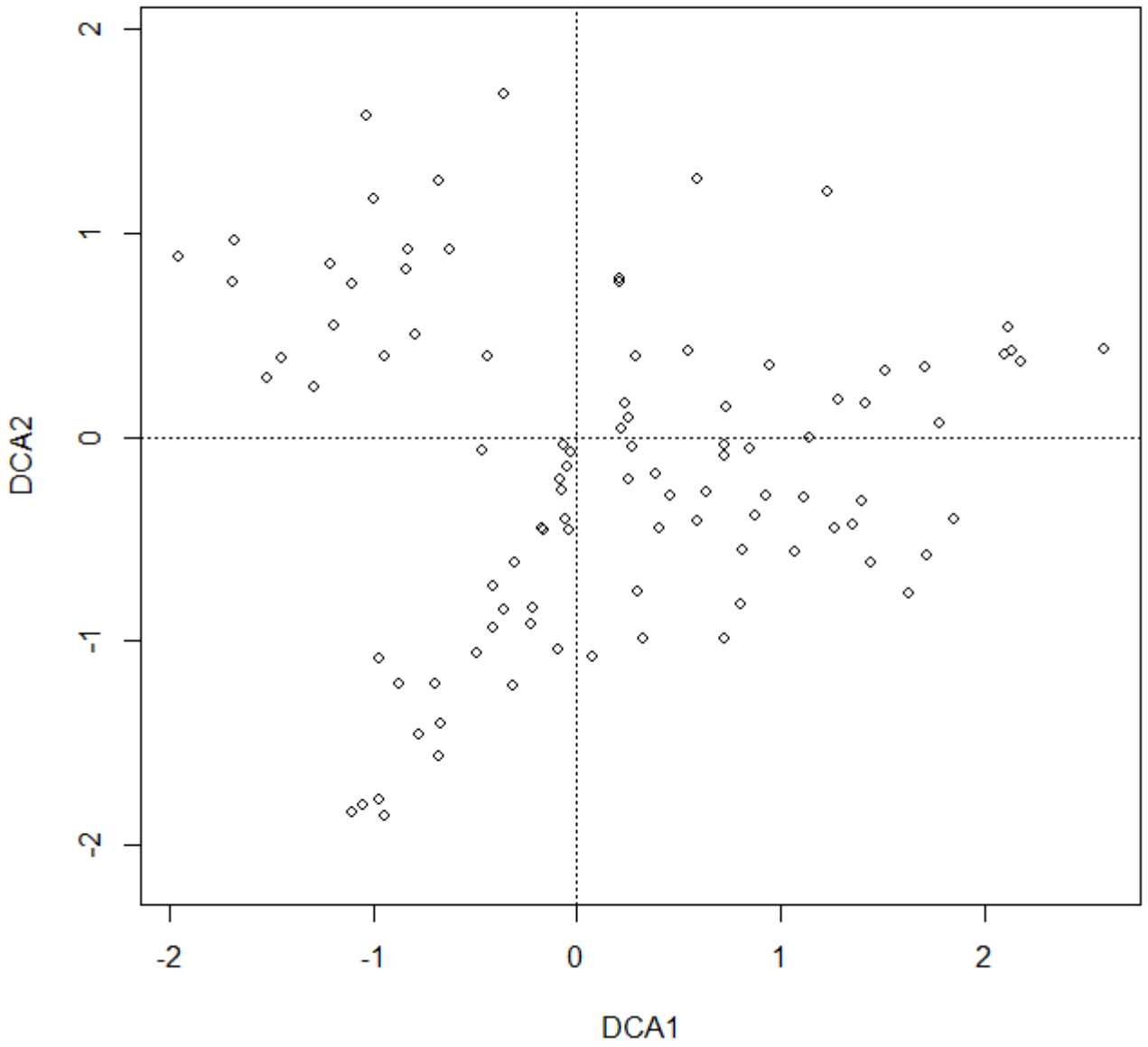
```
veg.data <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava-
spe.txt', row.names = 1)
env.data <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava-
env.txt')

library (vegan)
DCA <- decorana (veg = log1p (veg.data))
```

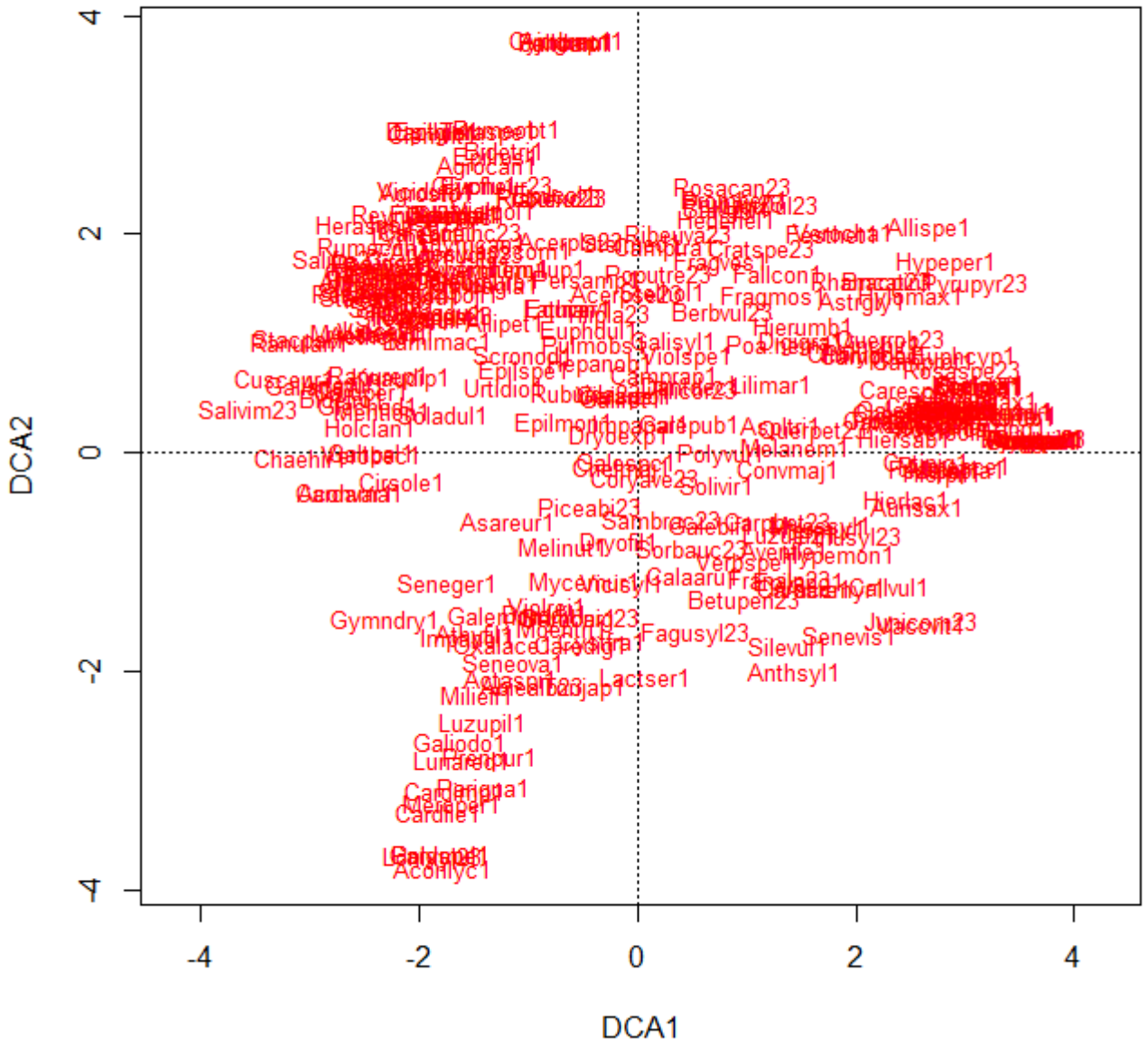
ordiplot (library vegan)

Draws ordination diagrams.

```
ordiplot (DCA, display = 'sites', type = 'p')
```



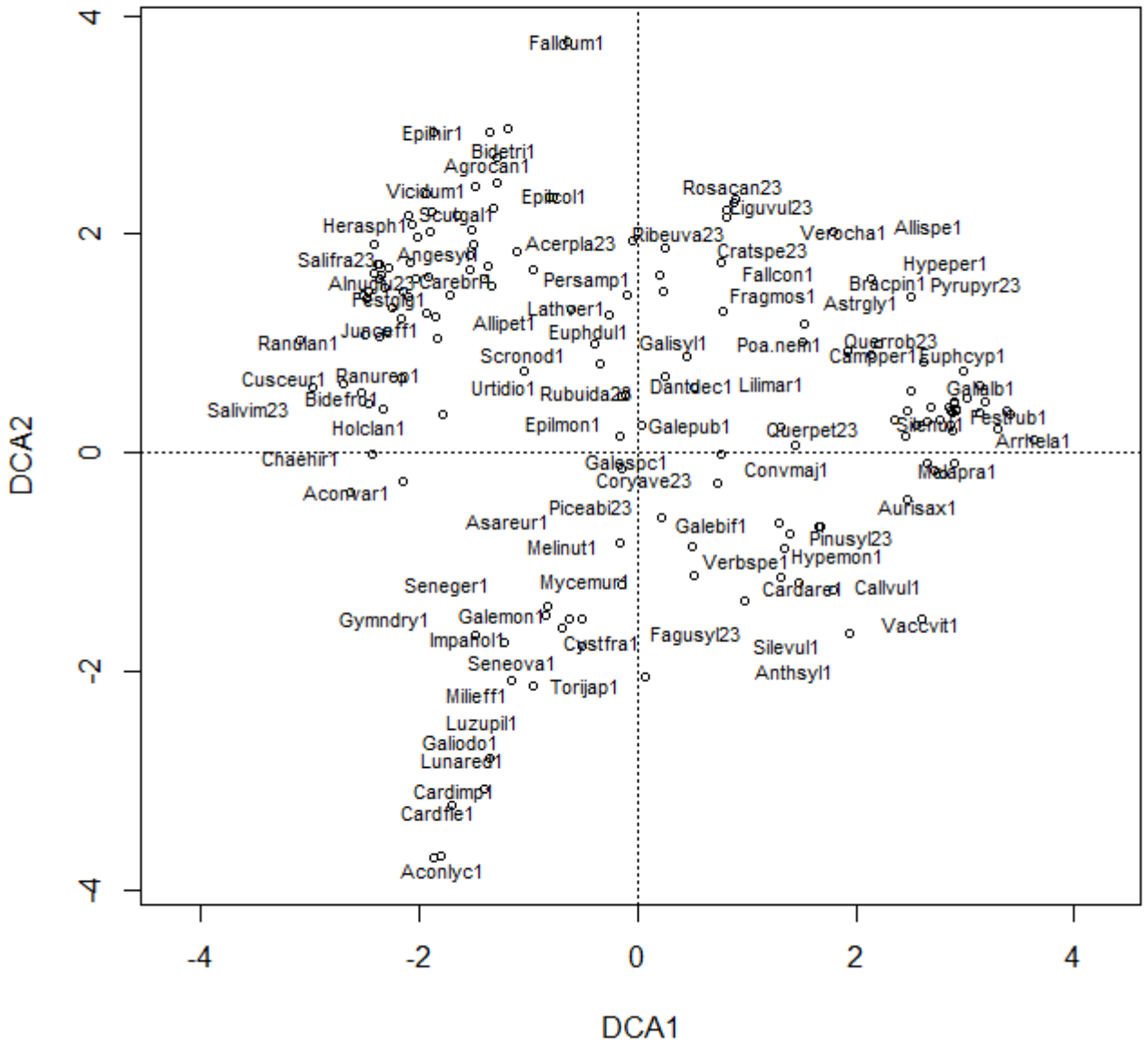

```
ordiplot (DCA, display = 'species', type = 't')
```



orditorp (library vegan)

Adds the labels onto an ordination diagram, so as they can be read (and don't overlap). First, draw empty diagram using function `ordiplot` and argument `type = 'n'`. Then add the labels using function `orditorp`.

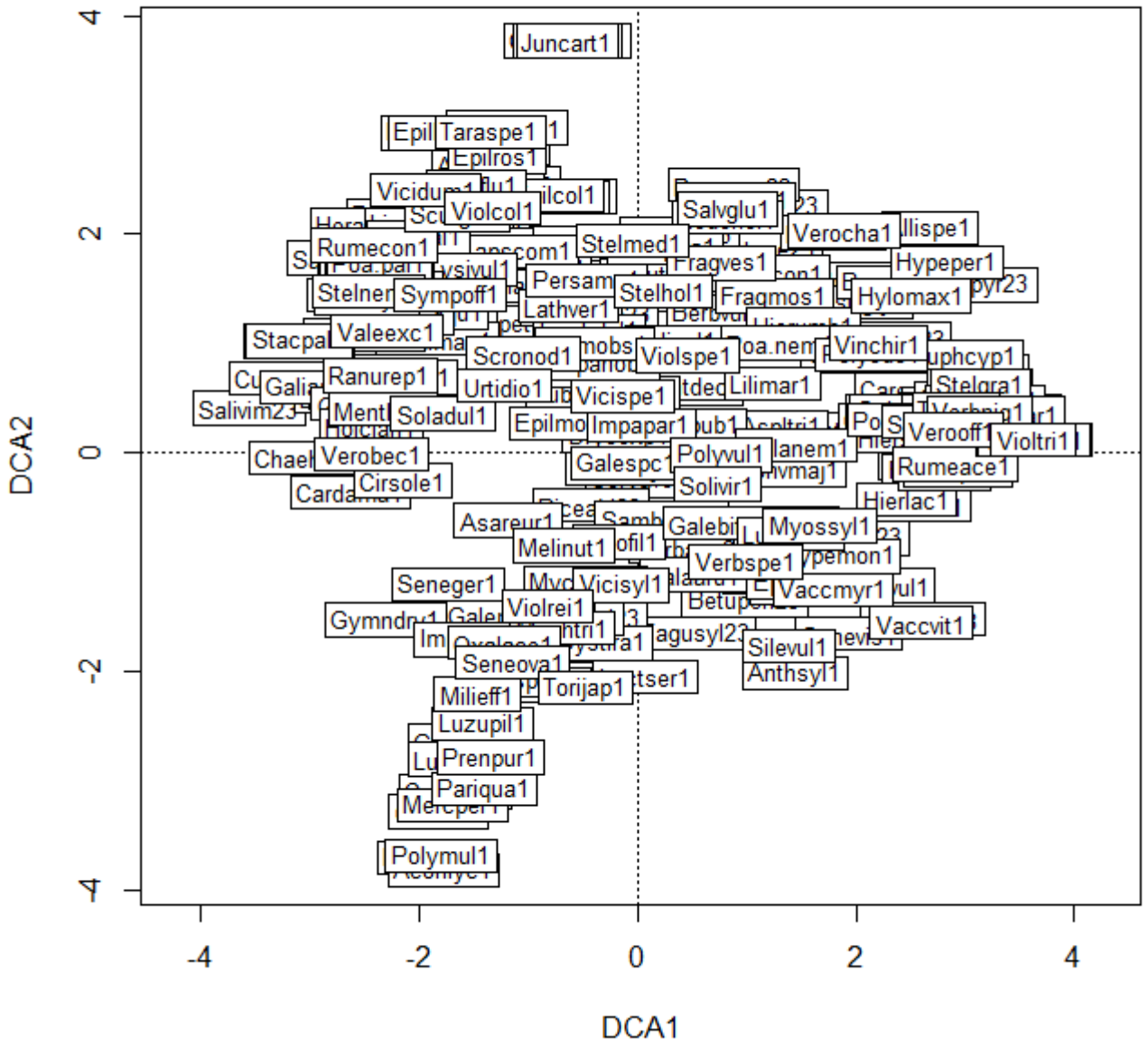
```
ordiplot (DCA, display = 'sp', type = 'n')
orditorp (DCA, display = 'sp')
```



ordilabel (library vegan)

Adds the labels which looks like stickers:

```
ordiplot (DCA, display = 'sp', type = 'n')
ordilabel (DCA, display = 'sp')
```

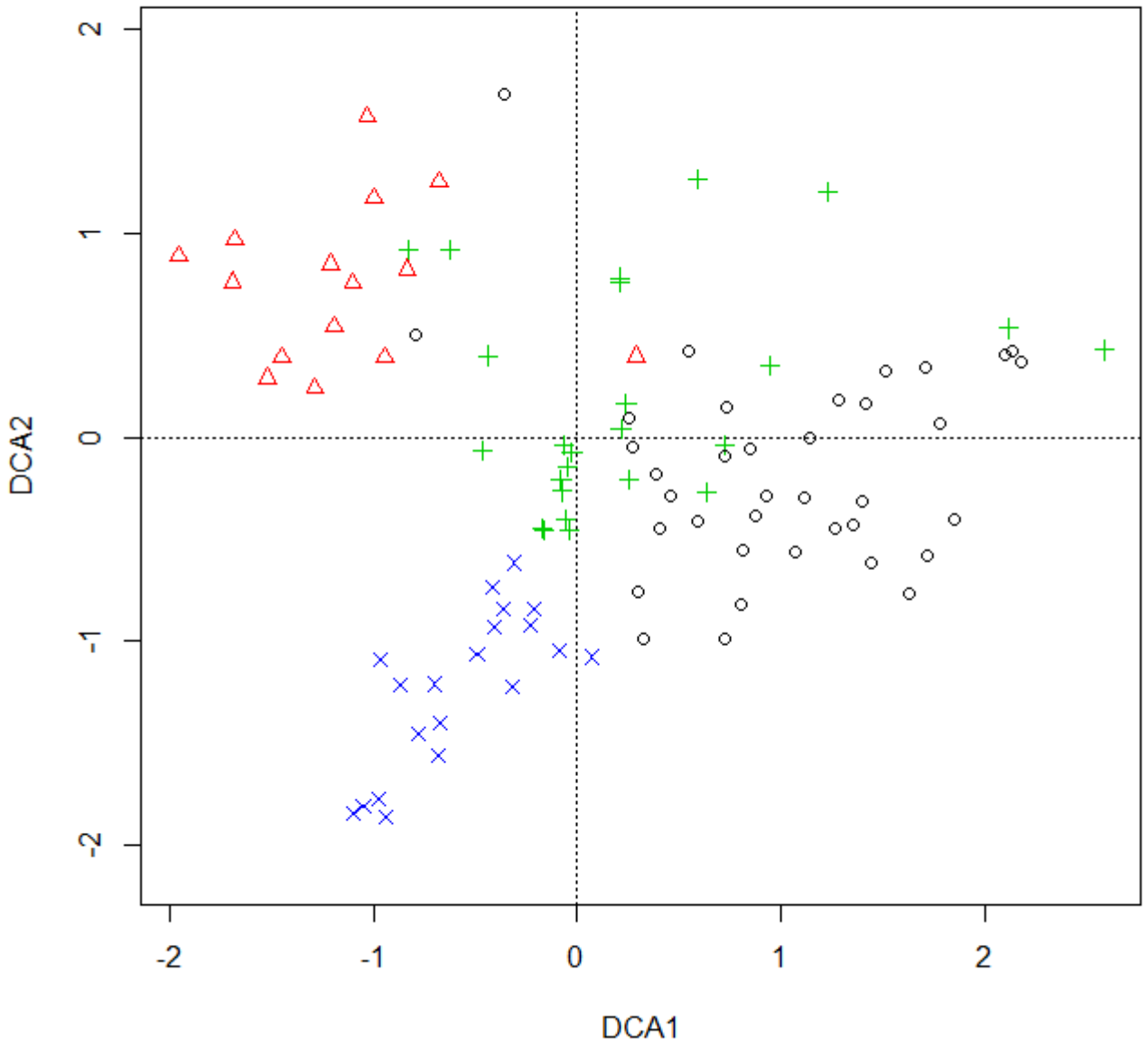


Other options: `identify.ordiplot` (adding the labels onto diagram by clicking the mouse), `orditklabel` (interactive diagrams in tcltk interface, but still not too elegant).

points (library vegan)

Adds points to ordination diagram - you can control their color, symbol, size etc. The following examples use the classification of samples (done by cluster analysis) into four groups, which is stored in variable GROUP in `env.data` data frame. First, draw empty ordination diagram (argument `type = 'n'` in function `ordiplot`) and add colourful points according to `env.data$GROUP` variable using function `points`:

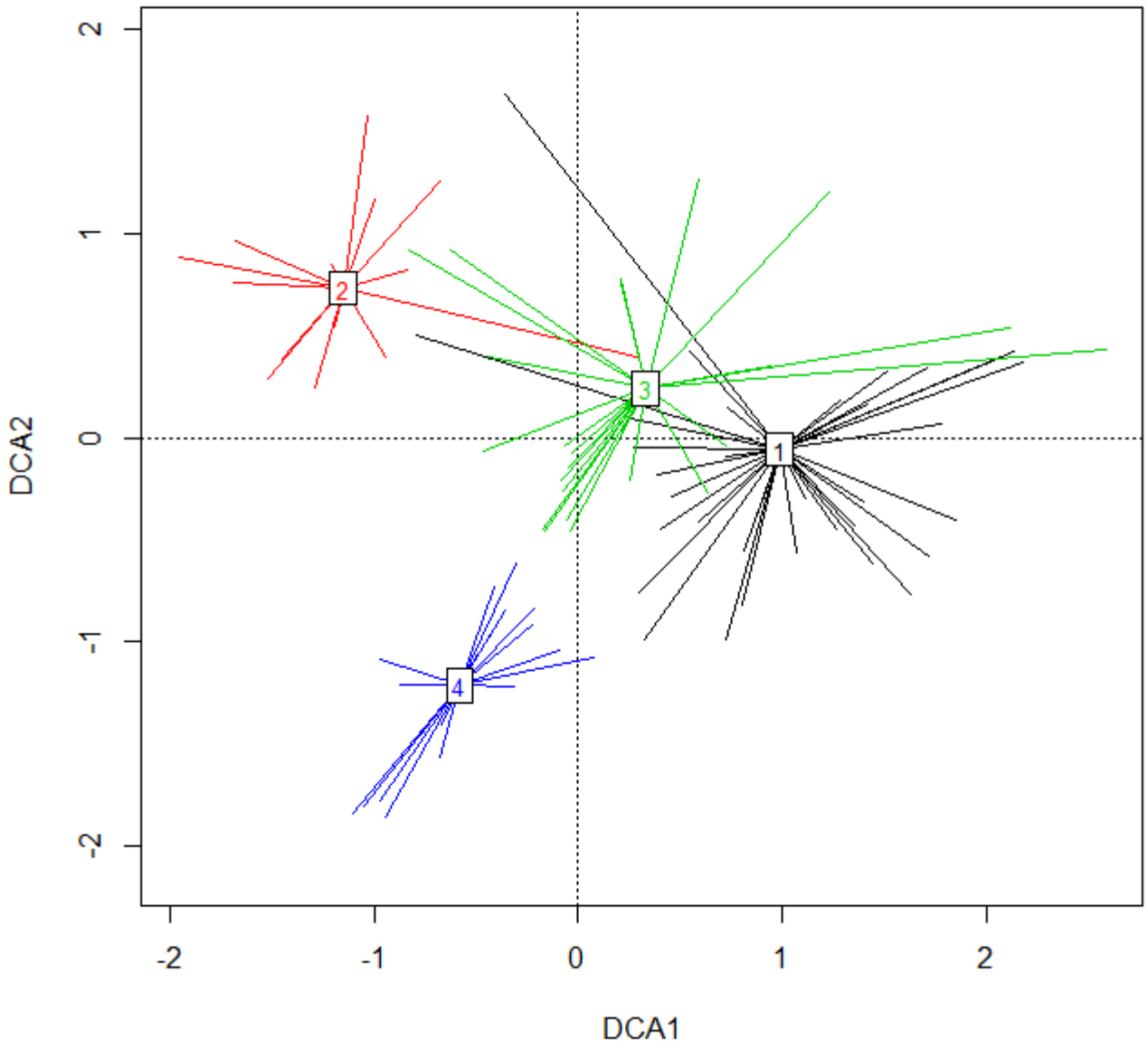
```
ordiplot (DCA, display = 'si', type = 'n')
points (DCA, col = env.data$GROUP, pch = env.data$GROUP )
```



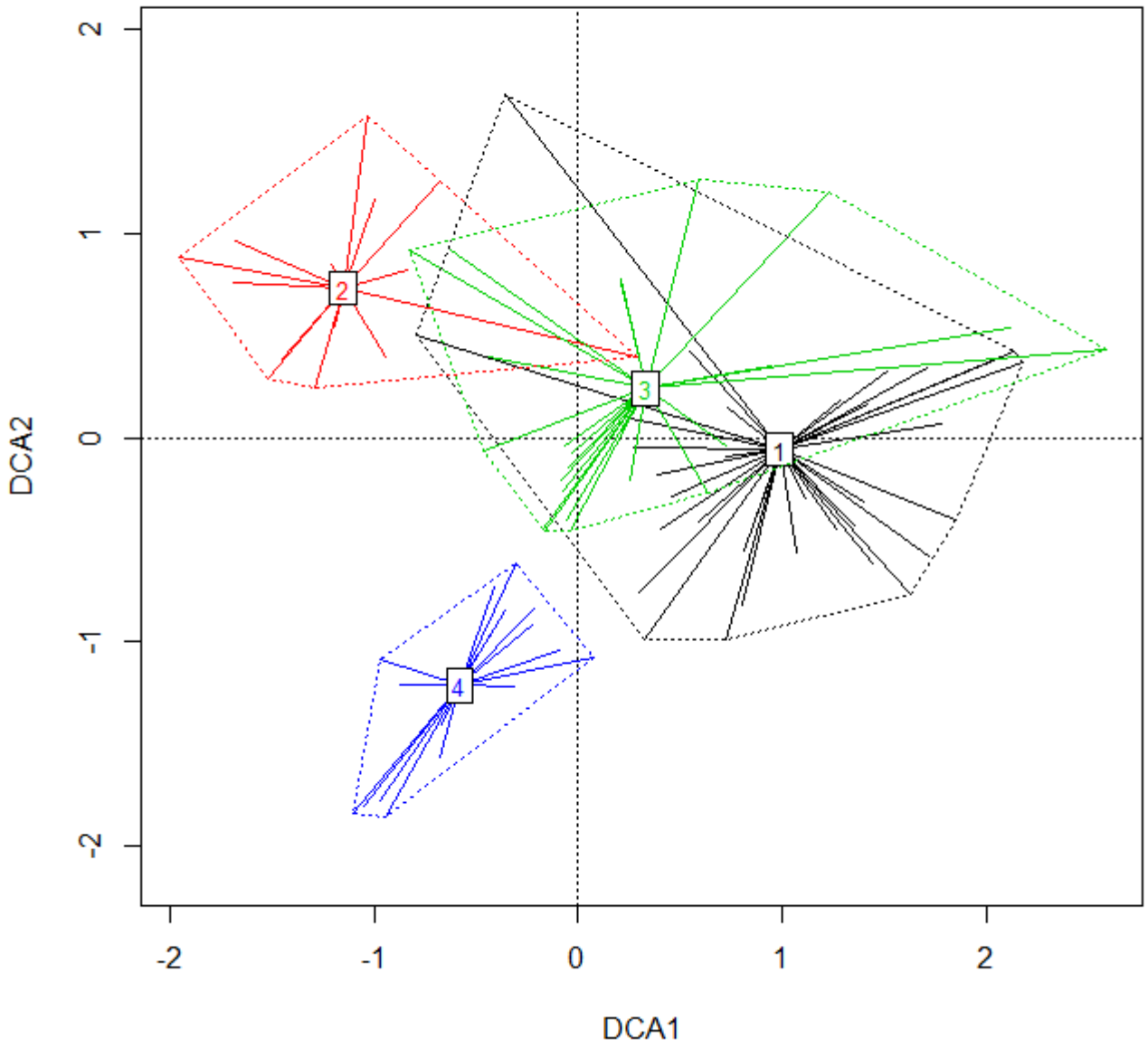
ordispider and ordihull (library vegan)

Function `ordihull` adds envelopes around groups of samples, while function `ordispider` adds so called spiders, connecting each sample with the centroid of particular group. Argument `label = T` draws the labels into the centroid of each spider:

```
ordiplot (DCA, display = 'si', type = 'n')
for (i in seq (1, 4)) ordispider (DCA, groups = env.data$GROUP, show.groups
= i, col = i, label = T)
```

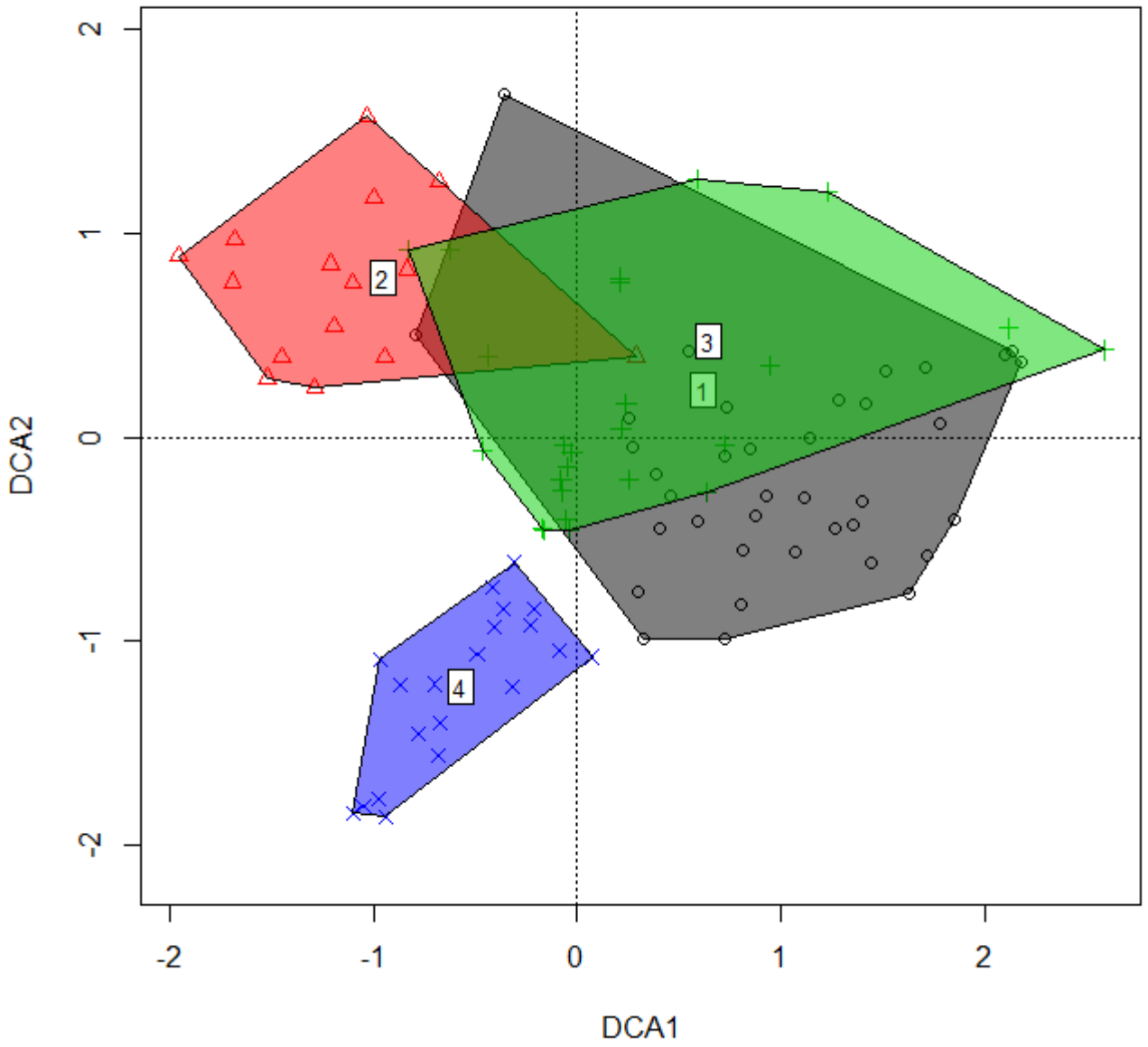



```
for (i in seq (1, 4)) ordihull (DCA, groups = env.data$GROUP, show.groups =
i, col = i, lty = 'dotted')
```



Function `ordihull` can draw not only envelopes, but also transparent polygons, by modifying the argument `draw = "polygon"` and optionally also transparency by argument `alpha` (default `alpha = 127` means that polygons will be semitransparent):

```
ordiplot (DCA, display = 'si', type = 'n')
points (DCA, col = env.data$GROUP, pch = env.data$GROUP)
for (i in unique (env.data$GROUP)) ordihull (DCA, groups = env.data$GROUP,
show.group = i, col = i, draw = 'polygon', label = T)
```



Note that setting `label = TRUE` also draws group labels, but this time not in the centroids of all the plots, but in centroids of plots which are part of envelope margin (that's why their position differs from spiderplot figure above)

ordicenter (custom function)

Adds labels into the group centroids. This can be usefull if you want to draw only the group centroids, not the envelopes or spiderplots (using `ordihull` or `ordispider`). Custom function (definition here), with code heavily borrowing from `ordispider` function in `vegan`, with the same arguments - check `?ordispider` for more details.

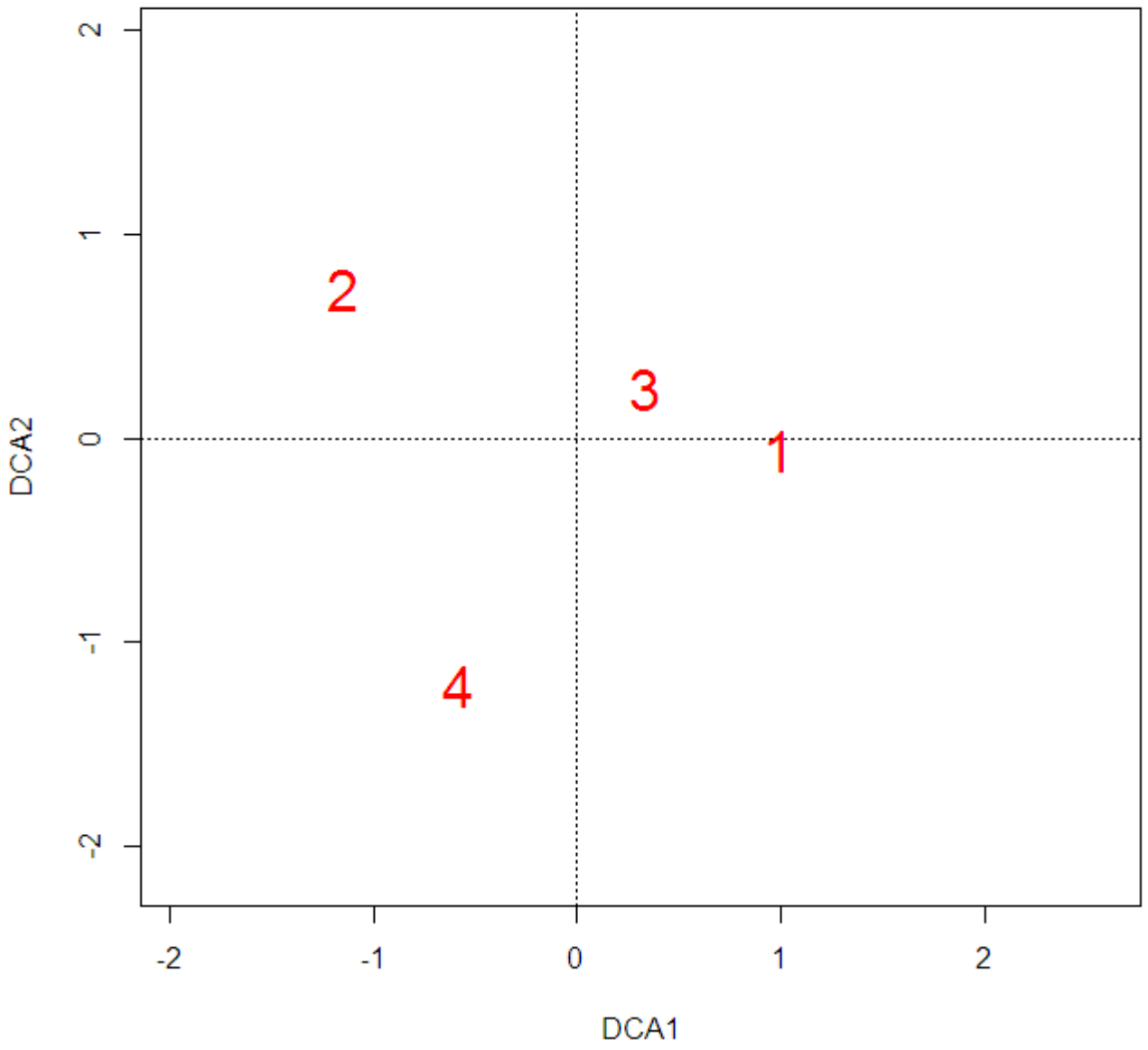
The function can be *sourced* from here in the following way:

```
source
```

```
('http://www.davidzeleny.net/anadat-r/doku.php/en:customized_functions:ordicenter?do=export_code&codeblock=0')
```

Draw plain ordination diagram with group numbers as centroids:

```
ordiplot (DCA, display = 'si', type = 'n')
ordicenter (DCA, groups = env.col = 'red', cex = 2)
```

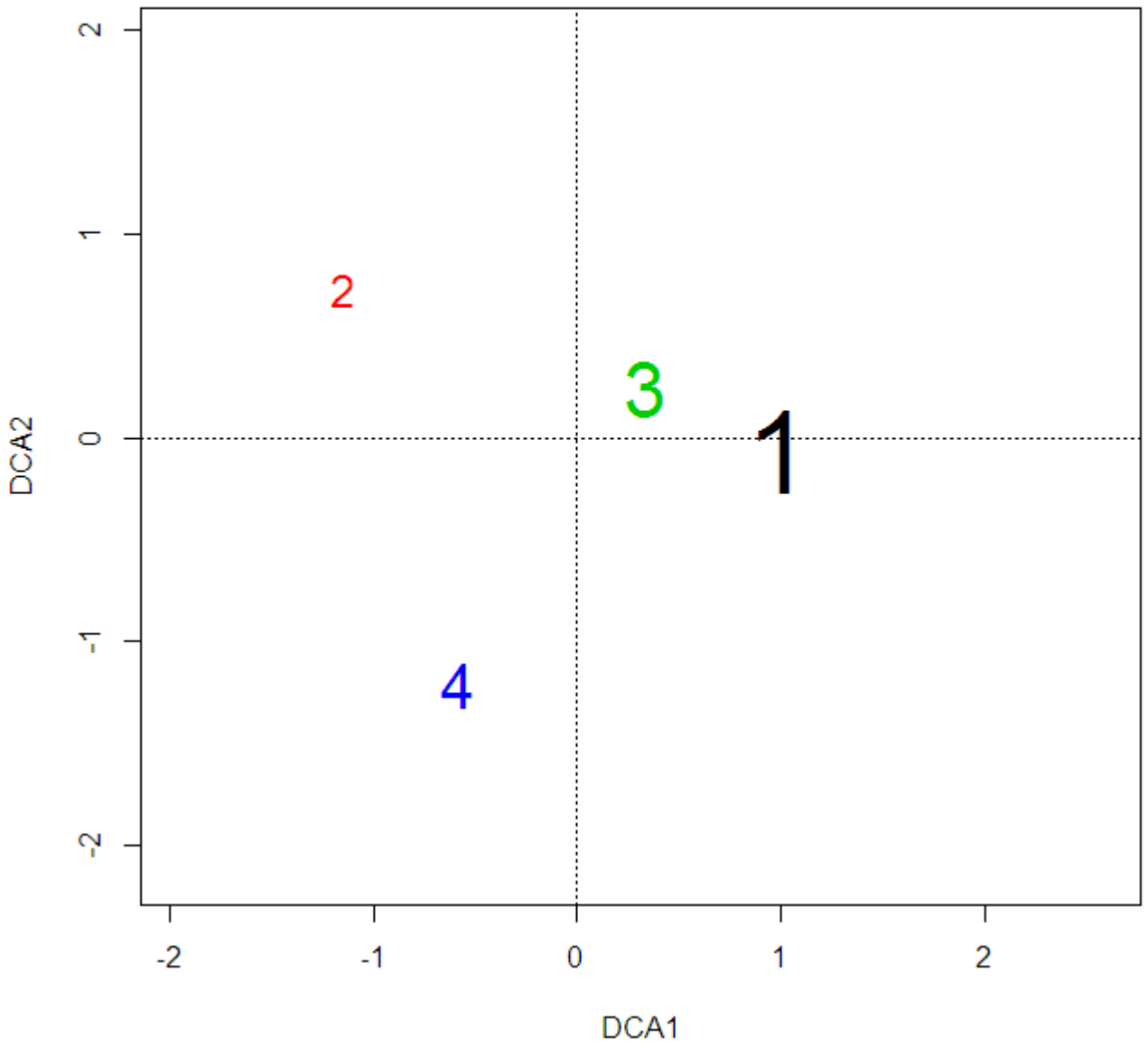


And a bit more advanced visualization - the size of the label is proportional to number of samples in the group, and labels differ by color:

```
ordiplot (DCA, display = 'si', type = 'n')
scaling.parameter <- as.vector (table (env.data$GROUP))/max (as.vector
(table (env.data$GROUP)))
for (i in 1:length (unique (env.data$GROUP)))
```

```
  ordicenter (DCA, groups = env.data$GROUP, show.groups = i, col = i, cex =
4*scaling.parameter[i])
```

(Note: the `scaling.parameter` simply calculates number of samples within each group (using function `table`, with output transormed into vector), and standardize these values between 0 and 1 by dividing them by the size of largest group. This scaling is in the next step (function `ordicenter`) used to multiply the maximum size (here 4) of the label in the argument `cex` to resize the centroid label.)

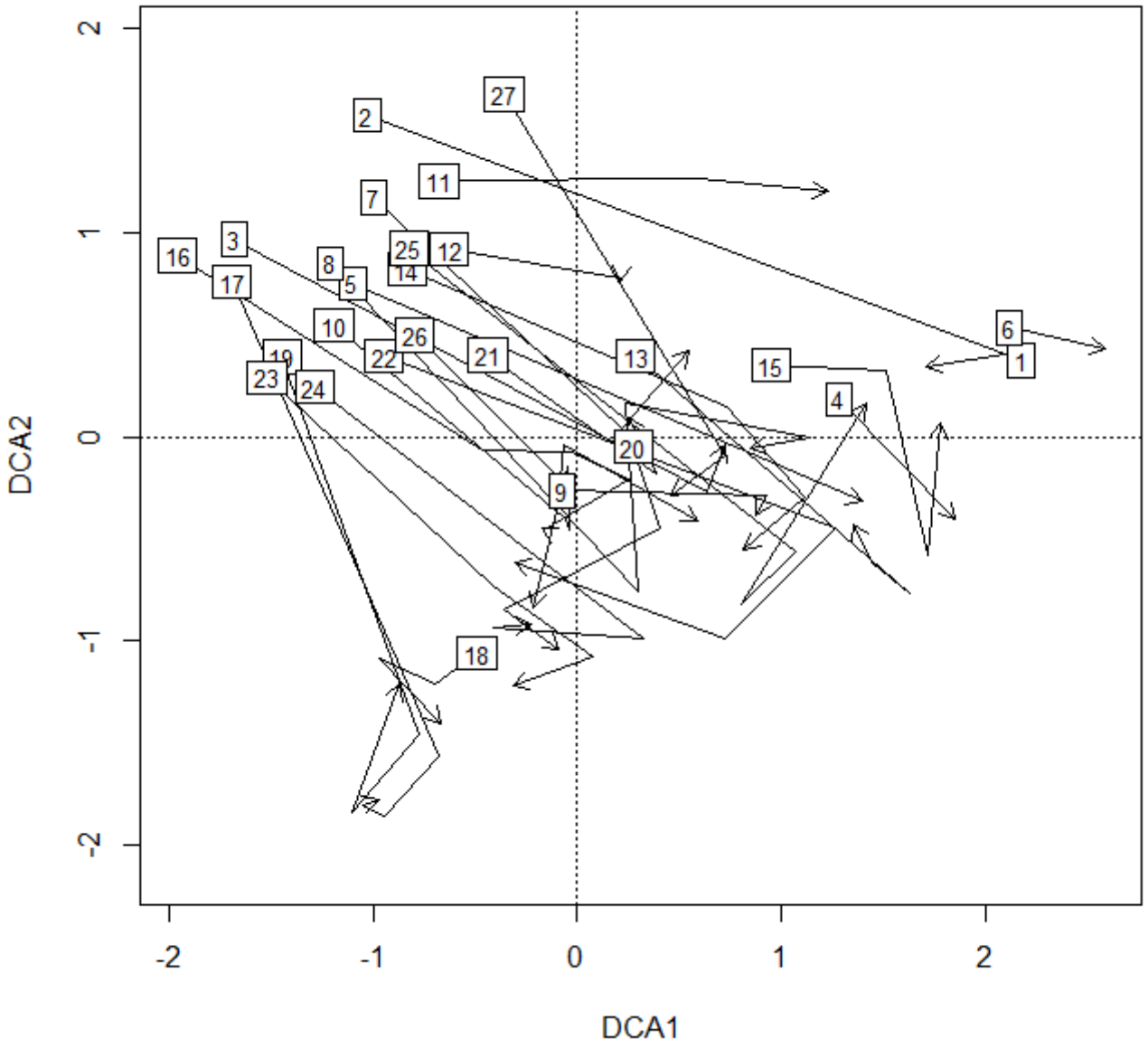


ordiarrows (library vegan)

This function adds the arrows connecting the samples in the certain order within a group. In the case of Vltava river dataset, this can be used to visualize the direction, in which the composition of vegetation changes along the transect from the bottom of the valley (wet alluvial forest) toward the upper part of the valley (either dry habitats on southern slopes or mesic habitats on northern slopes).

```
ordiplot (DCA, display = 'si', type = 'n')
ordiarrows (DCA, groups = env.data$TRANSECT, order.by = env.data$ELEVATION,
startmark = 1, label = TRUE, length = .1)
```

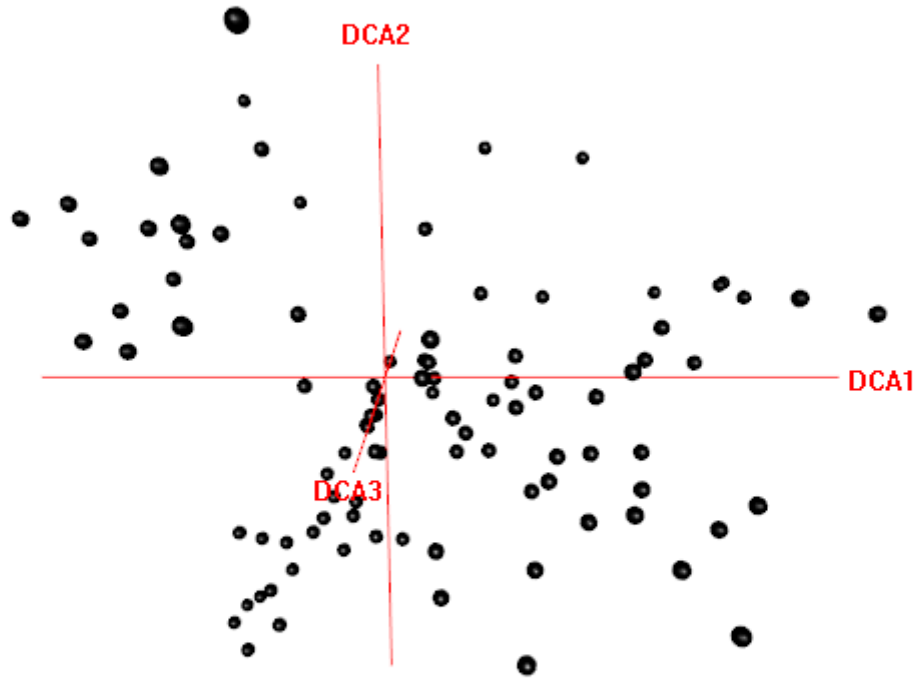
Note the meaning of the arguments (there are several other coding options how to specify which samples belongs to which arrow and how to sort the order of samples within the groups):`groups` specifies samples belonging to the same group, which will be drawn by one arrow; `order` specifies the variable used to define the order of samples how they will be drawn within the arrow[2)]; `label` defines whether the label with the group number should be drawn at the beginning of the arrow, and `length` is a specific argument which modified the behaviour of `arrows`, the underlying function actually drawing the arrows (namely it makes the length of the arrow head shorter).



ordirgl (libraries vegan3d and rgl)

This function creates three dimensional ordination diagrams, which can be rotated by clicking left mouse button and moving, and zoomed in and out by the mouse wheel[3)].

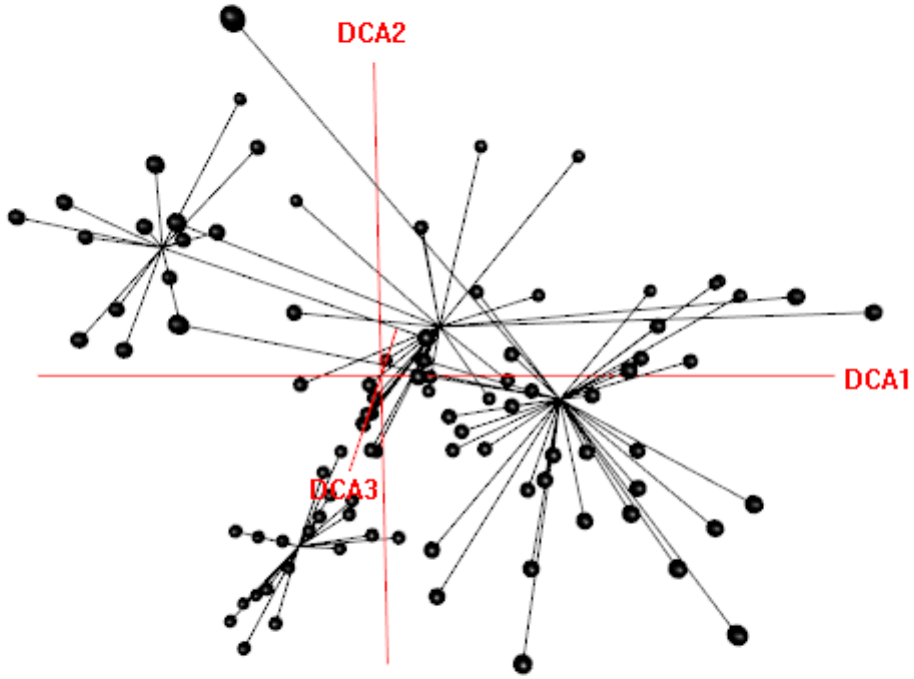
```
library (vegan3d)
ordirgl (DCA)
```



orglspider (library vegan3d and rgl)

3D spider plot.

```
orglspider (DCA, groups = env.data$GROUP)
```



orglhull (needs to be defined, requires library geometry)

A custom function I wrote couple of years ago, drawing the convex hulls around the groups of samples in ordination diagrams. You need to install library `geometry` first, if not yet done:

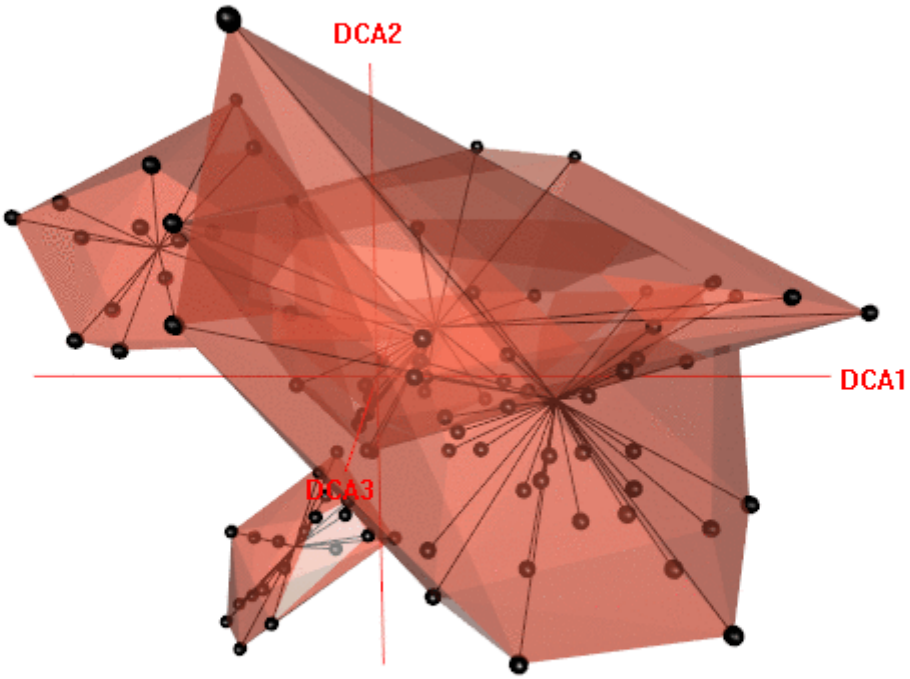
```
install.packages ('geometry')
```

The definition of the function (author D. Zelený, fix for R version 2.15.0 introduced by Paolo Piras) can be found here: orglhull, or you can directly copy this source script:

```
source
('http://www.davidzeleny.net/anadat-r/doku.php/en:customized_functions:orglhull?do=export_code&codeblock=1')
```

And here is how to use it:

```
orglhull (DCA, groups = env.data$GROUP, col = 'tomato', alpha = 0.5)
```



1)

But you will need to calculate the whole analysis in CANOCO.

2)

In the specific case of this dataset it is important, since the samples in the table are sorted according to the order they have been collected in the field; I started in the upper part of the valley and went down, making the vegetation plots; at the valley bottom, I have chosen locality of another transect and started to climb up, making the vegetation plots. This means that the plots sometimes start at the upper part of the transect and sometimes at the bottom.

3)

The GIF animations below are done using the `movie3d` function from `rgl` package. This functions requires to have ImageMagick installed on the computer. Detail script how these animations were done is here