Section: Numerical classification

# Cluster analysis (hierarchical agglomerative classification)

Theory R functions **Examples** Exercise

## Example 1: Comparison of hclust and agnes using Vltava river valley data

This exercise compares two main clustering functions, `hclust` and `agnes` (from library `cluster`), shows how they work and how they differ in use.

### hclust

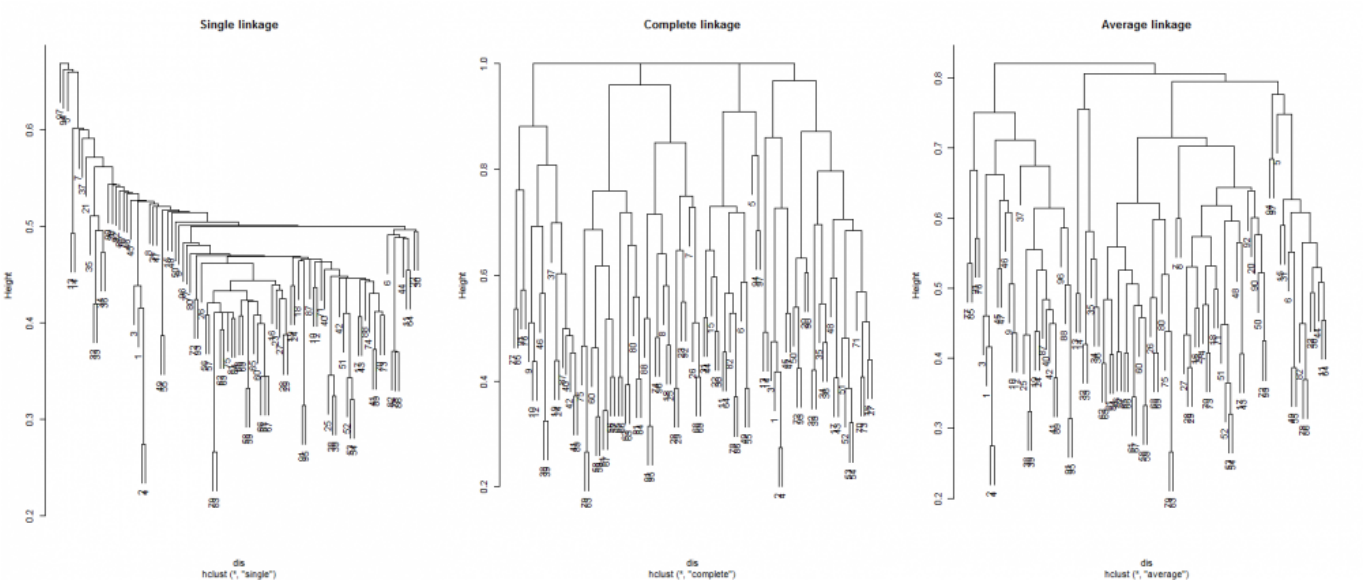
Requires two arguments: `d` - matrix of distances among samples (see Ecological resemblance), and `method` - name of clustering algorithm. Clustering algorithm are principally of three types: *single linkage*, *complete linkage* and *average linkage* - the third one is the one most often used in ecology (*average linkage* includes also popular *Ward method* or *beta flexible*). *Single linkage* method produce chained dendrograms, `complete linkage` dendrograms look a bit like rakes - some clusters merge together at the highest dissimilarity. `Ward method` is favourite, as it produces nicely compact clusters - note, however, that *Ward method* should not be combined with distance measures, which are not strictly metric, which is e.g. popular Bray-Curtis distance. *Beta flexible* method is popular, because setting beta coefficient allows one to modify the chaining of the dendrogram (*beta flexible* is not available in `hclust`, but in function `agnes` in the packages `cluster`).

```
vltava.spe <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vltava-
spe.txt', row.names = 1)
library (vegan)
dis <- vegdist (sqrt (vltava.spe), method = 'bray') # percentage cover data
are transformed by square root
cluster.single <- hclust (d = dis, method = 'single')
cluster.complete <- hclust (dis, 'complete')
cluster.average <- hclust (dis, 'average')
```

### plot.hclust

Draws the dendrogram.

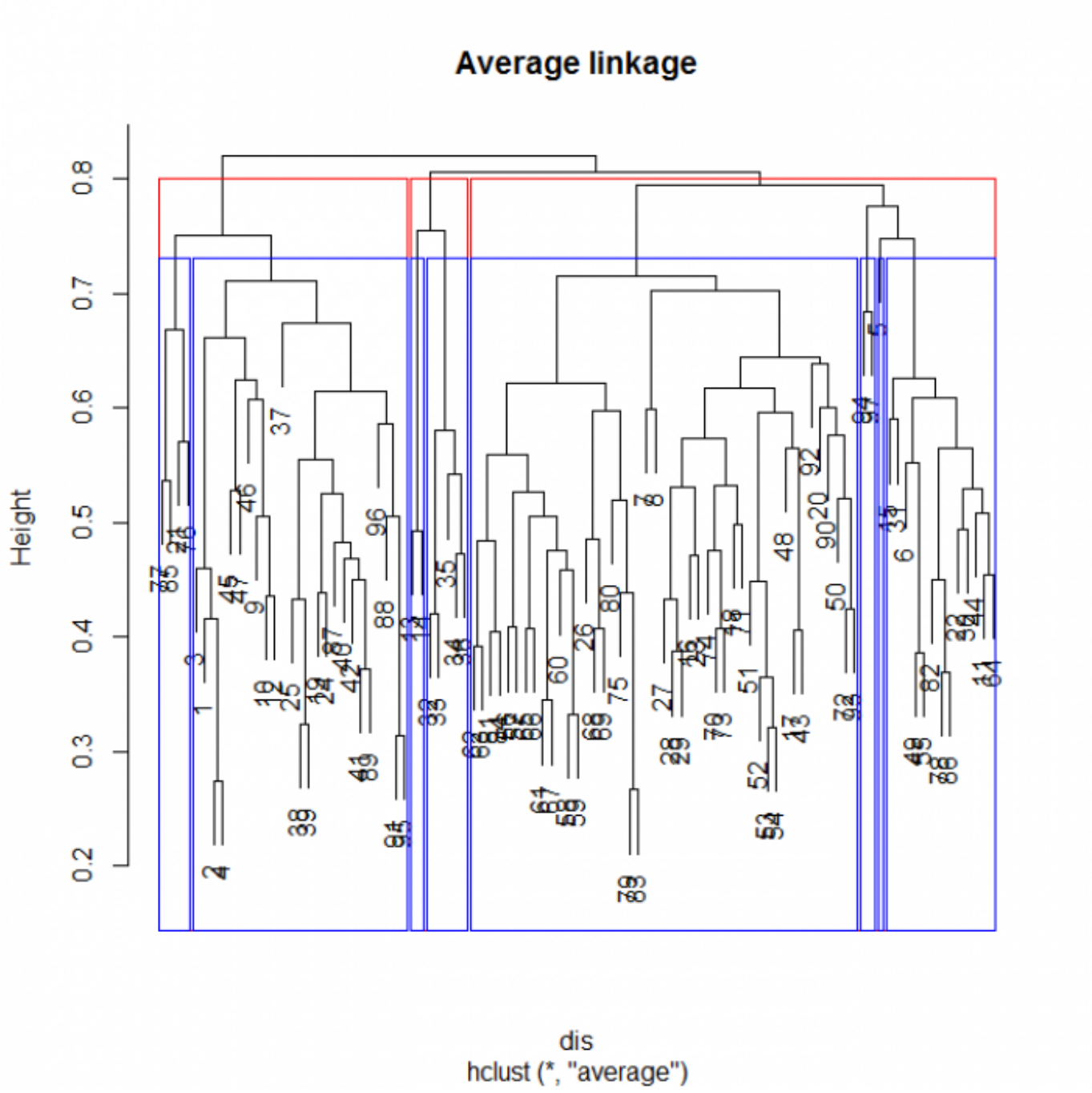
```
par (mfrow = c (1,3)) # will draw all dendrogram into one figure

plot (cluster.single, main = 'Single linkage')
plot (cluster.complete, main = 'Complete linkage')
plot (cluster.average, main = 'Average linkage')
```




### rect.hclust

Divides dendrogram into given number of groups (or in specified height, respectively)[1].

```
plot (cluster.average, main = 'Average linkage')
rect.hclust (cluster.average, k = 3)
rect.hclust (cluster.average, k = 8, border = 'blue') # argument border
specifies the colour of the rectangle
```




**cutree**

Returns a vector, containing assignment of samples into clusters.

```
clusters <- cutree (cluster.average, k = 5)
clusters
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 1  1  1  1  2  2  3  3  1  1  2  1  4  4  2  3  3  3  1  3  1  2  3
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
 1  1  3  3  3  3  2  2  4  4  4  4  4  1  1  1  1  1  1  3  2  1  1
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
 1  3  2  3  3  3  3  3  2  3  3  3  3  3  3  3  3  2  3  3  3  3  3
```

```
70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92
 3  3  3  3  3  3  1  1  2  3  3  3  2  3  3  1  2  1  1  1  3  1  3
93 94 95 96 97
 3  5  1  1  5
```

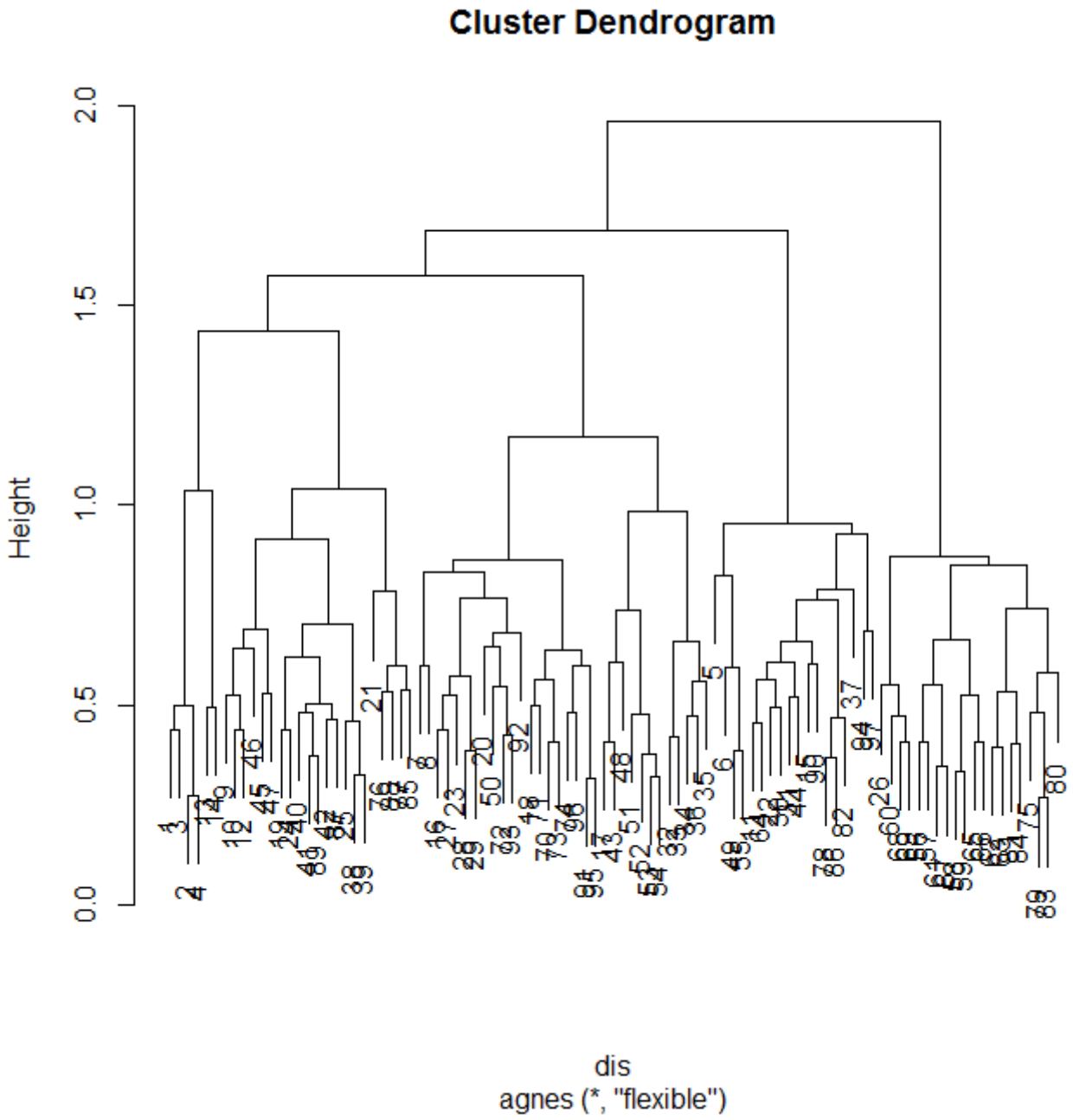
### agnes (library cluster)

Includes six clustering algorithms, some not included in function `hclust`. Important and interesting is the method `flexible`, known as *beta flexible*, which is often used for ecological data. Setting parameter beta influences the chaining of the dendrogram (for beta ~ +1 the chaining is maximal and result is similar to method *single linkage*, for beta = -1 is result similar to *complete linkage*). In function `agnes` the setting beta parameter is, however, not so simple: you need to set up one to four parameters (see `?agnes` for details). The simplest options is to assign to the argument `par.method` only one value, so called alpha, while it applies: beta = 1 - 2*alpha, the value for alpha can be thus calculated as alpha = (1-beta)/2. If, for example, I want to calculate *beta flexible* method with beta = -0.25 (which is said to optimally represent distances among samples), than alpha = (1-(-0.25))/2 = 1.25/2 = 0.625.

```
library (cluster)
cluster.flexible <- agnes (x = dis, method = 'flexible', par.method = 0.625)
```

Use `plot` function to draw the dendrogram (the function plots several figures, you need to click into the figure to move forward). If you want to use `plot.hclust` function, transform the `agnes` object into `hclust` object using `as.hclust` function.

```
cluster.flexible.hclust <- as.hclust (cluster.flexible)
plot (cluster.flexible.hclust)
```




## Example 2: Ward cluster algorithm applied on Barro Colorado Island data

In this example, we will use BCI dataset (data from tropical forest permanent plot) to conduct agglomerative cluster analysis, combining Ward's cluster algorithm with Bray-Curtis distance method. We will display resulting dendrogram with five distinguished vegetation groups, project their spatial configuration and also their composition differences in ordination diagram based on the same distance measure (NMDS using Bray-Curtis distances) and different distance measure (DCA based on chi-square distances).

First, let's load the data and log transform them (original data contain numbers of individuals):

```
library (vegan)
data (BCI)  # example using Baro Colorado data
BCI.log <- log1p (BCI)  # first, log transform species data, which contains
numbers of individuals
```

Function `vegdist` calculates distance matrix based on Bray-Curtis distances:

```
bc.dist <- vegdist (BCI.log, method = 'bray')
bc.dist
```

```
           1         2         3         4         5         6         7
8         9
2  0.2561624
3  0.2955959 0.2749839
4  0.3152203 0.2937687 0.2853559
5  0.3143658 0.3201234 0.3012306 0.2827513
6  0.3098983 0.3210060 0.3413611 0.3439200 0.3427188
7  0.3002090 0.3015008 0.3206233 0.3309095 0.3647759 0.2671652
8  0.3188611 0.2869892 0.2763554 0.2994629 0.3063330 0.3584762 0.3432045
9  0.3511634 0.3116517 0.2886386 0.3294022 0.3157618 0.3580870 0.3265907
0.2696480
10 0.3531552 0.3146364 0.2758954 0.2928692 0.3166512 0.3672627 0.3865676
0.2989141 0.2831257
...
```

Btw, if you type `bc.dist`, you actually use the function `print.dist`[2], which has several useful arguments (`?print.dist`). For example, if we want to print also diagonal values (zeros in this case, since we display distances, not similarites), we may use:

```
print (bc.dist, diag = TRUE)
```

```
           1         2         3         4         5         6         7
8         9
1  0.0000000
2  0.2561624 0.0000000
3  0.2955959 0.2749839 0.0000000
4  0.3152203 0.2937687 0.2853559 0.0000000
5  0.3143658 0.3201234 0.3012306 0.2827513 0.0000000
6  0.3098983 0.3210060 0.3413611 0.3439200 0.3427188 0.0000000
7  0.3002090 0.3015008 0.3206233 0.3309095 0.3647759 0.2671652 0.0000000
8  0.3188611 0.2869892 0.2763554 0.2994629 0.3063330 0.3584762 0.3432045
0.0000000
9  0.3511634 0.3116517 0.2886386 0.3294022 0.3157618 0.3580870 0.3265907
0.2696480 0.0000000
...
```

The Ward's algorithm is available in both the base function `hclust` and also the function `agnes` from the package `cluster`; the implementation, however, slightly differs. Murtagh & Legendre (2014) pointed out that in literature and software applications there are two versions of Ward's cluster algorithm, with only one being the correct implementation of the method originally published by Ward. The method `ward` in the function `agnes` contains the correct implementation, while default

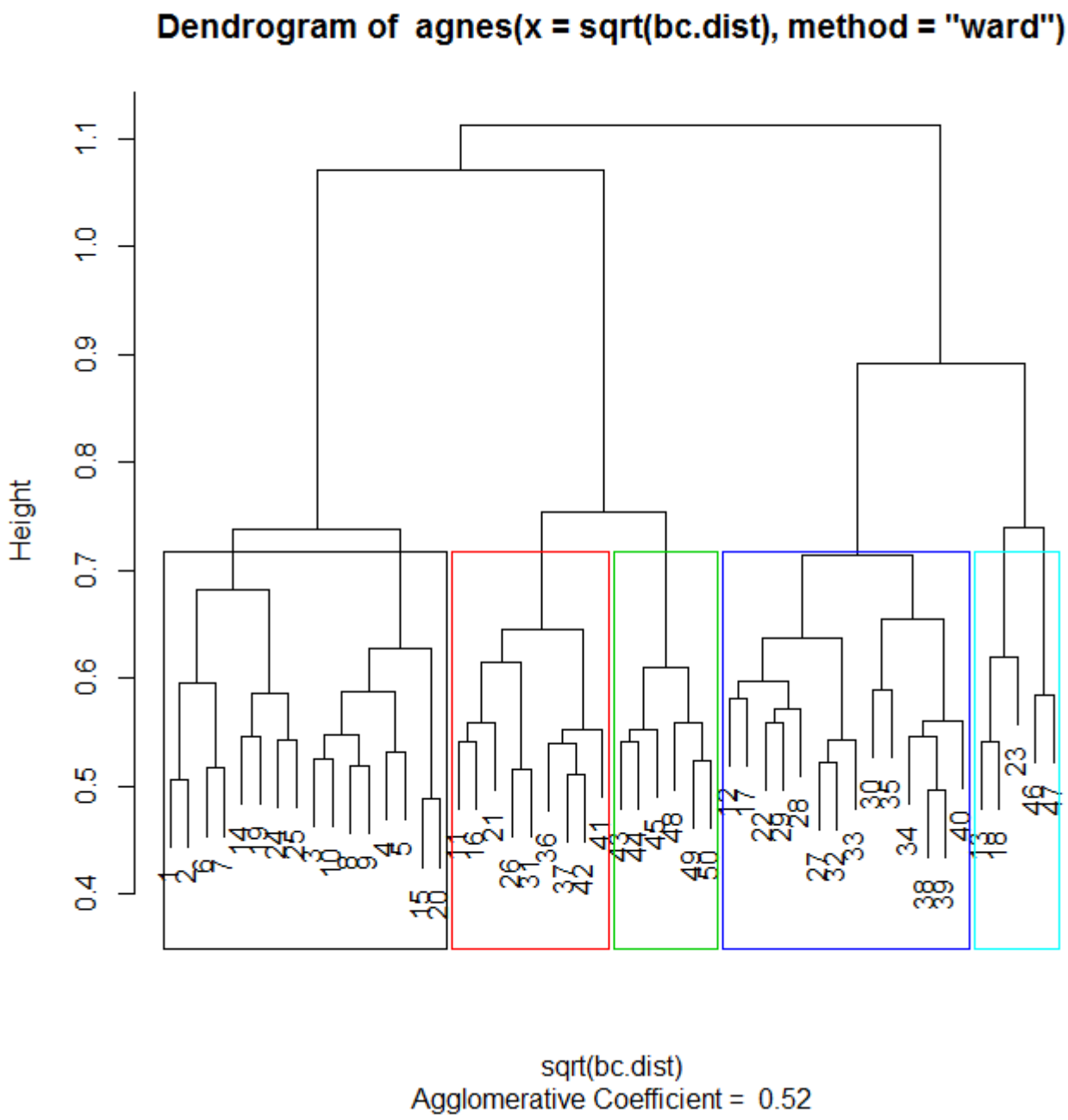
method `ward` in `hclust` does not (newly, this method is equal to `ward.D`); if `hclust` should calculate the correct Ward, the argument method need to be changed into `ward.D2`. Check `?hclust` to see the difference between both algorithm, and optionally search for more details in Murtagh & Legendre (2014).

Here, we will use `agnes` function from `cluster` package, with method set to `ward`. Since Ward algorithm requires distances to be metric (i.e. possible to display in metric Euclidean space) and Bray-Curtis is not metric, instead of raw Bray-Curtis distances I use their square-root transformation (see Ecological resemblance for details why).

```
#install.packages ('cluster') # install if necessary
library (cluster)
clust <- agnes (sqrt (bc.dist), method = 'ward') # calculate Ward's
algorithm
# on square-rooted Bray-Curtis distances
```

Resulting cluster dendrogram can be plotted using the function `plot`. Note that the object `clust` is of the class `agnes`, so `plot.agnes` will be used in case that generic `plot` function is applied on it; in case that it is the result of `hclust`, the function `plot.hclust` would be used. The function `plot.agnes` by default plots two figures and works in interactive mode (needs to hit enter in order to proceed to the next one (the second one is the dendrogram). By adding argument `which.plot = 2`, only the dendrogram will be plotted (for meaning of `which.plot`, check `?plot.agnes`).

```
plot (clust, which = 2)
rect.hclust (clust, 5, border = 1:5)
```



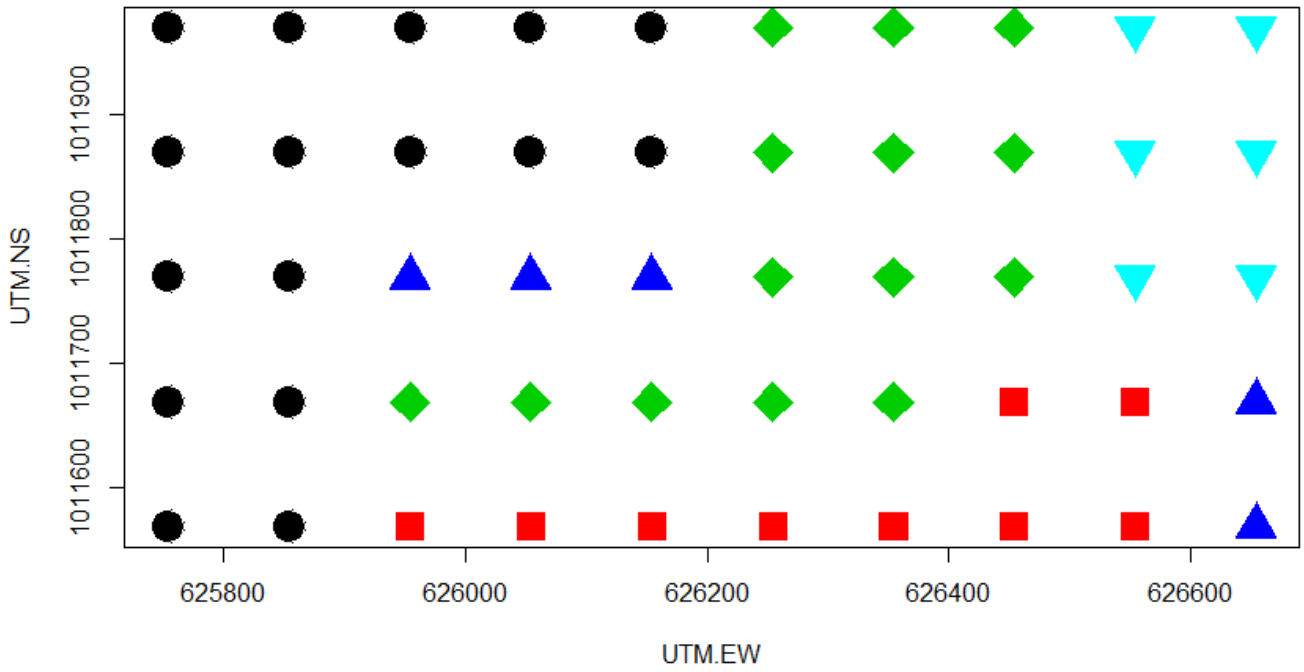

Finally, the function `cutree` (sounds like *to cut tree*, with only one *t*) can cut the dendrogram according to predefined criteria, either into given number of clusters (argument k), or according to given level of similarity/dissimilarity (argument h). Here, we will cut the dendrogram into four groups of samples:

```
groups <- cutree (clust, k = 5)
groups
```

```
 [1] 1 1 1 1 1 1 1 1 1 2 3 4 1 1 2 3 4 1 1 2 3 4 1 1 2 3 3 3 3 2 3 3 3
[35] 3 2 2 3 3 3 2 2 2 2 2 5 5 2 2 2
```

The next step is to draw the spatial distribution of plots classified into four clusters. For this, we need spatial coordinates of each plot, which are in the dataset `BCI.env` as coordinates `UTM.NS` (latitude) and `UTM.EW` (longitude):

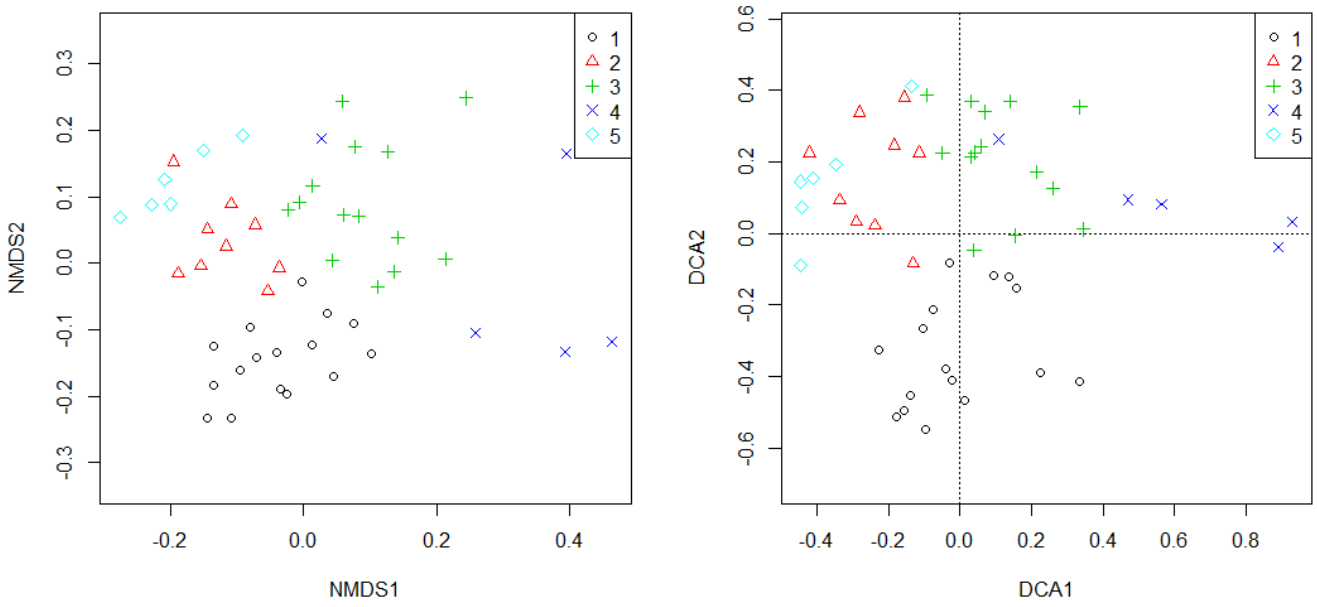
```
BCI.env <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/bci.env
.txt')
plot (UTM.NS ~ UTM.EW, data = BCI.env, pch = groups, cex = 3)  # this is the
simple version, only with symbols differentiating individual groups
plot (UTM.NS ~ UTM.EW, data = BCI.env, pch = groups+20, cex = 3, bg =
groups, col = 'white') # this is more "colorful" option. Note that symbols
now are 21 to 25, arguments 'bg' and 'col' which
```




Result of cluster analysis can be displayed also onto the ordination diagram of unconstrained ordination, to asses how well are individual groups compositionally separated from each other. Here we will use two contrasting unconstrained ordination analyses: one which is based on the same distance measure as is the cluster anlaysis (NMDS with Bray-Curtis distance), one which is not (DCA, whose mother method, CA, is based on chi-square distance). The point of this selection (NMDS vs DCA) is to show that to evaluate whether cluster analysis results into well defined clusters, one needs to use the ordination method which is based on the same distance metric as the cluster analysis (Bray-Curtis distance in this example).

```
# First, NMDS with Bray-Curtis distances
NMDS <- metaMDS (vegdist (BCI.log))  # note that I could use also "NMDS <-
metaMDS (bc.dist)" here
par (mfrow = c(1,2)) # I want to plot both plots into one figure, with two
panels in one row
ordiplot (NMDS, type = 'n')
points (NMDS, pch = groups, col = groups)
legend ('topright', pch = 1:5, col = 1:5, legend = 1:5, bty = 'n')

# Second, DCA ordination (implicitly using chi-square distance)
```

```
DCA <- decorana (BCI.log)
ordiplot (DCA, type = 'n', display = 'si')
points (DCA, pch = groups, col = groups)
legend ('topright', pch = 1:5, col = 1:5, legend = 1:5, bty = 'n')
```




The difference in the distribution of individual groups between ordinations is not too big in case of this dataset. But remember that the best for projecting results of cluster analysis is to use the same distance measure (here Bray-Curtis) for both cluster analysis and ordination.

1)

For alternative way, how to distinguish the clusters by color, see function `ColorDendrogram` in package `sparcl`.

2)

When applied, the script contains only `print` function, which is a generic function choosing the specialized function by asking the class of the object on which it is applied; in case of `bc.dist`, the object is of the class `dist`, so the generic function `print` will choose function `print.dist` to proceed. Check `?print.dist` to see the helpfile for this function.