# 4. Hierarchical Cluster Analysis

**FISH 560: Applied Multivariate Statistics for Ecologists**

**Topics**

- Perform hierarchical agglomerative clustering, plot dendrograms

**R Libraries:** vegan, cluster, pvclust, NbClust, clusteval
**R Source:** biostats

---

**BACKGROUND**

Cluster analysis encompasses several multivariate techniques that are used to group objects into categories based on their (dis)similarities. The aim is both to minimize within-group variation and maximize between-group variation in order to reveal well-defined categories of objects, and therefore reduce the dimensionality of the data set to a few groups of rows (James and McCulloch 1990, Legendre and Legendre 1998). Therefore, this approach is generally recommended when distinct discontinuities instead of continuous differences (i.e. gradients) are expected between samples (objects) because cluster analysis mostly aims at representing partitions in a data set.

Cluster analysis of a data matrix proceeds in two steps. First, a relevant association coefficient must be chosen to measure the association (similarity or dissimilarity) among objects or among variables. Second, the calculated association matrix is represented as a horizontal dendrogram or tree (hierarchical clustering) or as distinct groups of objects (k-means clustering), based on specific rules to aggregate objects. For ecologists, the power of cluster analysis derives from the existence of different types of (dis)similarity coefficients. The choice of appropriate and ecologically meaningful association coefficients is particularly important because it directly affects the values that are subsequently used for the categorization of objects.

Two general approaches to clustering are commonly used: hierarchical clustering and non-hierarchical partitioning. In hierarchical clustering, a linkage rule to form clusters and the numbers of clusters that best suit the data have to be determined *a priori* or subjectively based on the clustering results. Clusters, which are nested rather than mutually exclusive here, are formed by either agglomerative or divisive algorithms. The basic theory of agglomerative hierarchical analysis is to partition a similarity/distance matrix of n samples (or variables) into a hierarchy of k (k < n) groups using a clustering strategy. It begins with each object or variable, assigns these objects or variables to a cluster, and agglomerates them in a hierarchy of larger and larger clusters until finally a single cluster contains all the objects or variables. The five commonly used clustering techniques are single-linkage, complete-linkage, centroid clustering, average-linkage, and the minimum-variance clustering. Divisive cluster analysis begins with all objects together in a single cluster and successively divides the objects into a hierarchy of smaller clusters, either using a single variable (i.e. monothetic divisive cluster analysis) or many variables (i.e. polythetic divisive cluster analysis). These two strategies do not necessarily yield the same clusters.

---

**SET-UP**

In this exercise you will be working with the MAHA environment dataset and species abundance dataset.

Let's first set-up your R work session by defining the current work directory to your folder of choice and loading the vegan, cluster and pvclust libraries. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Second, import the MAHA environment dataset and MAHA species abundance dataset, by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

---

## CLUSTER ANALYSIS IN R

The cluster package in R includes a wide spectrum of methods, corresponding to those presented in Kaufman and Rousseeuw (1990). Curiously, the methods all have the names of women that are derived from the names of the methods themselves. Of the partitioning methods, **pam** is based on partitioning around mediods, **clara** is for clustering large applications, and **fanny** uses fuzzy analysis clustering. Of the hierarchical methods, **agnes** uses agglomerative nesting, **diana** is based on divisive analysis, and **mona** is based on monothetic analysis of binary variables. Other functions include **daisy**, which calculates dissimilarity matrices, including Gower's coefficient. Basic agglomerative hierarchical clustering can be performed with **hclust** in the Stats library and the **agnes** function in the Cluster library. The **hclust** function will be the main focus of this chapter.

---

## CONDUCT HIERARCHICAL AGGLOMERATIVE CLUSTERING

Hierarchical clustering is typically used when the objective is to establish groups of similar objects and simultaneously describe the relationships among the groups. The results are typically depicted in a dendrogram, which is a tree-like plot depicting the agglomerative or divisive sequence of fusions or splits, respectively. Each step in cluster analysis is described below.

### 1. Compute an appropriate dissimilarity/distance matrix

A necessary pre-cursor to any cluster analysis is to compute an appropriate dissimilarity/distance matrix (although sometimes this step is included in the clustering algorithm and therefore does not need to be done separately). For example, let's calculate the Euclidean distance based on site environmental characteristics (see R documentation: Ecological Resemblance for the entire list of association coefficients). First, let's standardize the environment matrix (mean = 0, variance = 1; i.e., z-score) in order to control for the fact that variables are measured on different scales (recall the nasty properties of Euclidean distance), by typing:

```
envdata.tra<-data.stand(envdata,method='standardize',margin='column',plot=F)
```

How might you quickly double-check that the function actually transformed your data appropriately? Hint: type **mean(envdata.tra[,1])**. Now, let's calculate the Euclidean distance matrix, by typing:

```
site.eucd<-vegdist(envdata.tra,method='euclidean')
```

### 2. Compute hierarchical clustering

Now we are ready to build the hierarchy using one of two basic strategies: agglomerative and divisive; and many alternatives within each of these general approaches. Here, we'll focus on the clustering function from the stats package (this package is automatically loaded when R is opened). The function performs hierarchical agglomerative clustering on dissimilarity matrices supplied by the user.

Its usage is:

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

The arguments are:

- d – the dissimilarity matrix to be analyzed
- method – the agglomeration method to be used. Choices are:

    - ward.D2 – Implements Ward's (1963) minimum variance method, compared to ward.D which does not (see Murtagh and Legendre 2014).
    - single – Single linkage.
    - complete – Complete linkage. The default method.
    - average
    - mcquitty
    - median
    - centroid
- members – vector assigning sample units to clusters. Optional; used to construct a dendrogram beginning in the middle of the cluster analysis (i.e., beginning with the clusters already specified by this argument).

Like always, you can type "**`?hclust`**" at the **R** prompt for additional documentation.

Try typing any one of the following commands:

```
sitecl.ave<-hclust(site.eucd,method='average')
sitecl.cen<-hclust(site.eucd,method='centroid')
sitecl.med<-hclust(site.eucd,method='median')
sitecl.sin<-hclust(site.eucd,method='single')
sitecl.com<-hclust(site.eucd,method='complete')
sitecl.ward<-hclust(site.eucd,method='ward.D2')
```

**3. Examining clustering results**

The results of the clustering process are given in the corresponding object. For the object produced by hclust(), we can print a simple summary of the agglomeration sequence using the hclus.table() function in biostats, by typing:

```
hclus.table(sitecl.ave)
```

The table gives the sequence of fusions and the corresponding distance at which the fusion took place. In the agglomeration sequence, if a number j in the row is negative, then the single observation |j| is merged at this stage. If j is positive, then the merger is with the cluster formed at stage j of the algorithm. The Euclidean distance between objects/clusters is also provided. Here is some of the output:

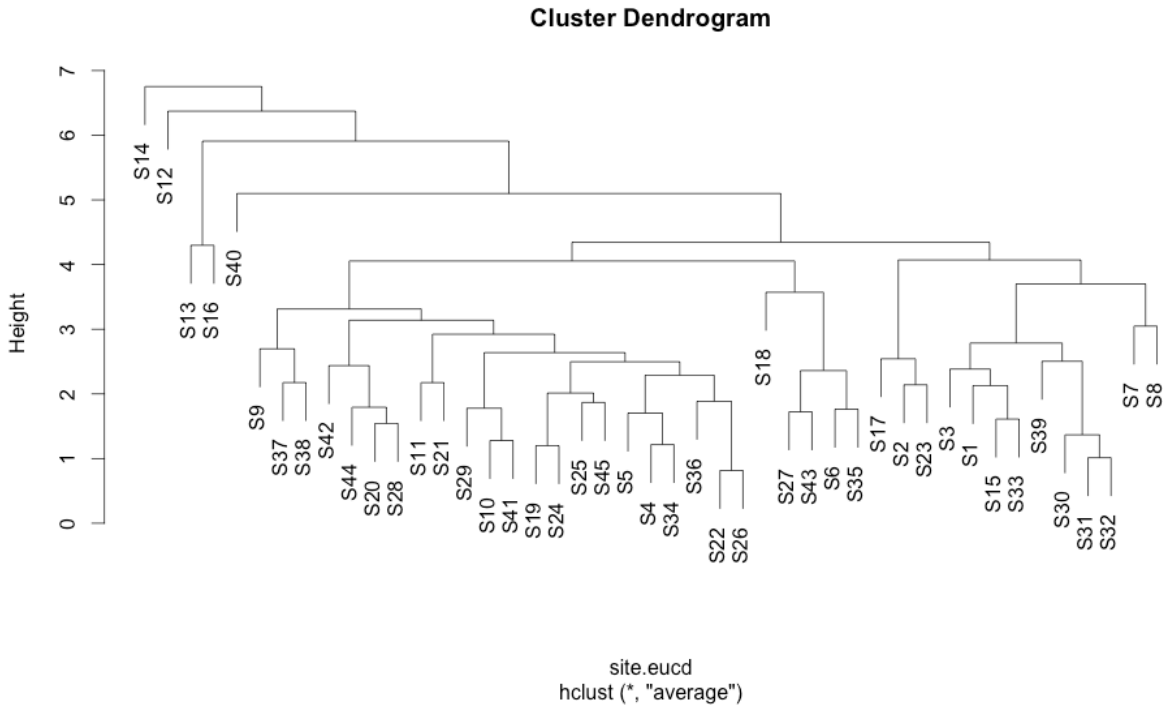
```
$dist.method
[1] "euclidean"

$method
[1] "average"

$cluster.table
   no. clusters entity entity  distance
1            44    -22    -26 0.8159428
2            43    -31    -32 1.0138844
3            42    -19    -24 1.2000233
4            41     -4    -34 1.2166423
5            40    -10    -41 1.2790520
…
```

**Dendrogram:** The primary way to inspect the results from hierarchical clustering is using a dendrogram plot. The dendrogram depicts the fusion or splitting sequence used to create the cluster hierarchy and graphically portrays the relationships both within and among clusters by showing the distance at which

objects fuse together or split apart. To create a dendrogram from the average-linkage agglomerative clustering produced by the hclust() function, type:

```
plot(sitecl.ave)
```

Your dendrogram should look identical to the one rendered below.



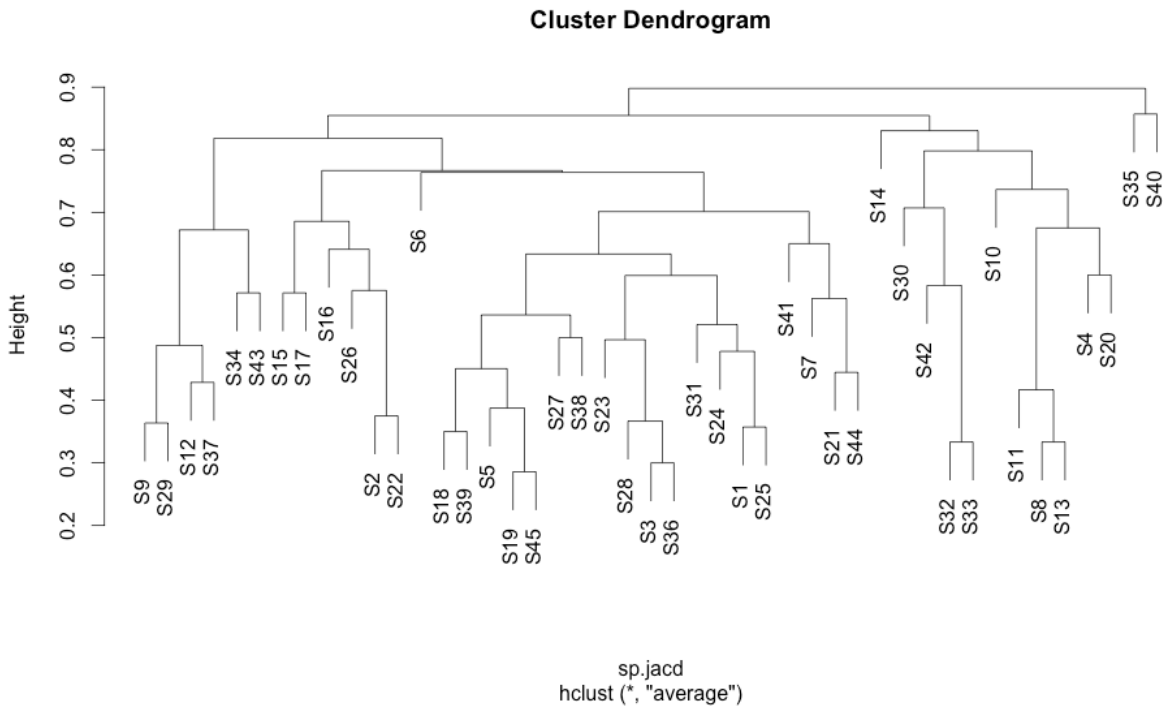

You can make your dendrogram fancier by typing:

```
plot(sitecl.ave,main='Average-linkage
Dendrogram',xlab='Sites',ylab='Euclidean distance',hang=-1)
```

Quickly, let's cluster the sites according to fish species presence/absence based on Jaccard's similarity coefficient! Type the following:

```
speocc <- data.trans(speabu,method='power',exp=0,plot=F)
sp.jacd<-vegdist(speocc,method="jaccard")
specl.ave<-hclust(sp.jacd,method='average')
plot(specl.ave)
```

You should be able to follow each of the steps above. You will obtain the following for the average-linkage agglomerative clustering.




Okay, let's focus again on the site-environment analysis. Based on the dendrogram, we may be ready to decide on the number clusters to retain for our final solution. We can show on the dendrogram the final cluster membership using the rect.hclust() function, either on the basis of the height of the dendrogram (i.e., dissimilarity value) or a specified number of clusters, as follows:

But first, let's re-plot the dendrogram of environmental distances based on average linkage.

```
plot(sitecl.ave)
rect.hclust(sitecl.ave,k=9)
```

This identifies 9 clusters in the dendrogram.

```
plot(sitecl.ave)
rect.hclust(sitecl.ave,h=6)
```

This identifies clusters in the dendrogram according to Euclidean distance = 6. That is, all objects must be within a Euclidean distance of 6 to be considered members of the same cluster.

Based on this analysis we can use the cutree() function to "cut" the dendrogram into clusters, by typing:

```
sitecl.class<-cutree(sitecl.ave,k=9)
```

This cuts the dendrogram into 9 clusters.

Note, the cutree() creates a vector called **`sitecl.class`** that contains the cluster membership (i.e., cluster ID) for each observation in the data set, which we will use below. Neat!

## 4. Cluster Diagnostics: Evaluating the cluster solution

**Agglomerative coefficient:** The agglomerative coefficient measures the amount of clustering structure of the dataset (see Lecture Notes). If observations quickly agglomerate into distinct clusters that later agglomerate into a single cluster at much greater dissimilarities, the coefficient will approach 1. By contrast, datasets with weak or no clustering will have an agglomerative coefficient approaching zero.

The agglomerative coefficient can be computed by typing:

```
coef.hclust(sitecl.ave)
```

This should return a value of 0.668. You can substitute any of the suitable cluster objects that you computed above.

**Cophenetic correlation coefficient:** One way to evaluate how well the cluster hierarchy represents the original object-by-object dissimilarity space is to compute the cophenetic correlation coefficient. The cophenetic distance between two observations that have been clustered is defined to be the intergroup dissimilarity at which the two observations are first combined into a single cluster (this is produced by the cophenetic function used below). The cophenetic correlation is the correlation between the dissimilarities in the original p-dimensional space and the cophenetic distances. It can be argued that a dendrogram is an appropriate summary of some data if the correlation between the original distances and the cophenetic distances is high. Otherwise, it should simply be viewed as the description of the output of the clustering algorithm.

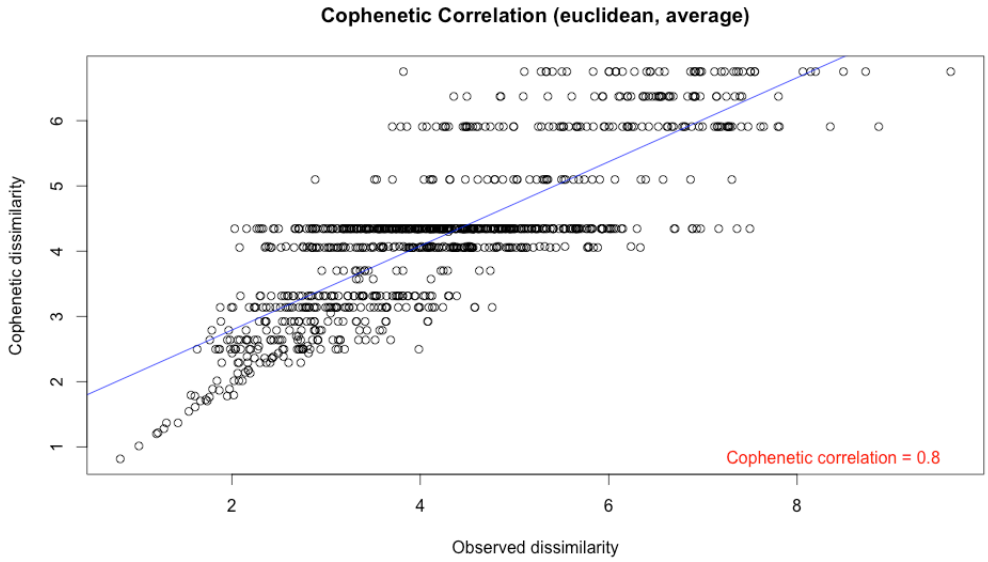
The cophenetic correlation can be computed directly by typing:

```
cor(site.eucd,cophenetic(sitecl.ave))
```

This should return a value of 0.803. Alternatively, we can plot the cophenetic relationship using the hclus.cophenetic() function in biostats by typing:

```
hclus.cophenetic(site.eucd, sitecl.ave)
```

This will also return a value or 0.80 and produce the following:

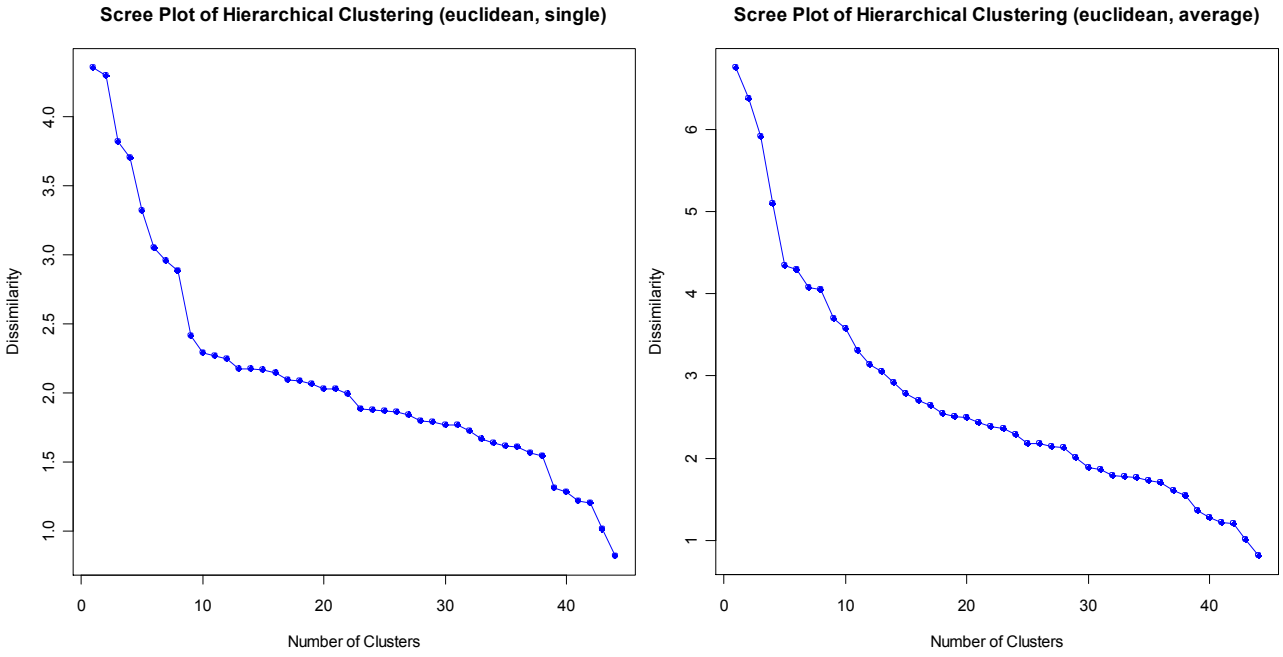


## 5. Cluster Diagnostics: Determining the number and stability of clusters

**Scree plot:** An important way to summarize the hierarchical structure of the clusters and determine the number of clusters is using a scree plot. A scree plot depicts the dissimilarity value of the fusion/split against the number of clusters. For an agglomerative hierarchical clustering, as the number of clusters increases, the dissimilarity value at which clusters fuse together typically decreases monotonically. In this case, the scree curve is read from right to left. Abrupt changes in the scree slope or a pronounced "elbow" in the scree plot is an indication of where there is a large change in dissimilarity as clusters fuse together. This is a logical level at which to "cut" the dendrogram. Let's look at the scree plot for clustering based on single linkage and average linkage, by typing:

```
hclus.scree(sitecl.sin)
hclus.scree(sitecl.ave)
```

You should obtain the following:



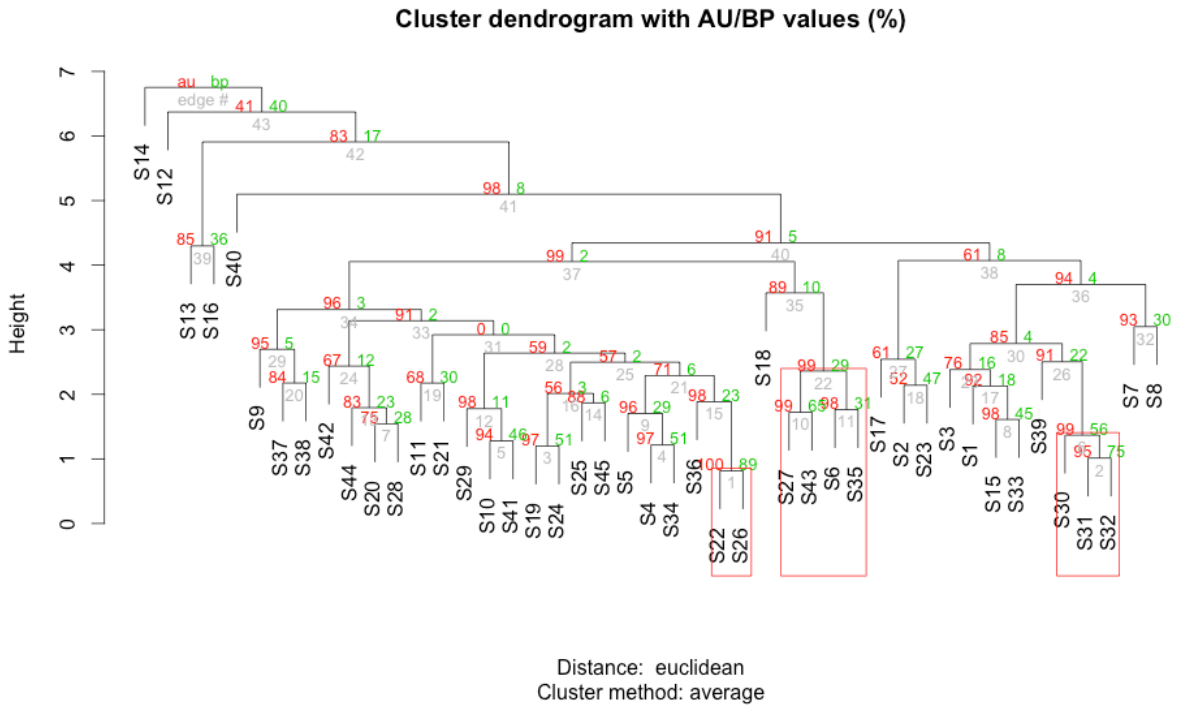

**Bootstrap Test:** It is always recommended that you evaluate the stability of the derived clusters. By doing so one can ask: Are the clusters cohesive? Do they remain intact with minor perturbations to the data? For example, if we select a subset of objects at random, do the original clusters remain intact, albeit reduced in size depending on which observations made it into the random subset? If we create a bootstrap sample from the original data set by sampling observations at random with replacement, do the original clusters remain intact in the bootstrap samples? Many such functions have been developed for DNA data analysis. Here, we will use the pvclust() function in the pvclust library to address these questions. This function calculates p-values for hierarchical clustering via multiscale bootstrap resampling. Begin by typing:

```
clus.stab<-pvclust(t(envdata.tra),method.hclust="average",method.dist
="euclidean",nboot=50)
```

Take note that we perform this computation on the tranpose of the environmental data matrix because this function assumes that objects are columns NOT rows. We have selected average-linkage clustering based on euclidean distance, and only run 50 bootstrap samples to save computational time. Next we can view the plot and highlight significant clusters at a critical P <0.01, by typing:

```
plot(clus.stab)
pvrect(clus.stab,alpha=0.99)
```

Based on the 50 random bootstrap samples the following dendrogram was produced with statistically significant clusters in red boxes (gray in your handout). The values at each node represent two types of p-values: au (Approximately Unbiased) p-value on the left and bp (Bootstrap Probability) value on the right. The AU p-value, which is computed by multiscale bootstrap resampling, is a better approximation to an unbiased p-value than the BP value, which is computed by normal bootstrap resampling. Note that the AU and BP values, as well as the statistical significance boxes, that you produce will be different that the values in the dendrogram below because your analysis will be based on a different set of bootstrapped samples. Also note that this method might not work for smaller sample sizes – just be warned!




**Index-based Assessment:** A number of indices are available to select the "optimal" number of clusters in a database. NbClust package provides 30 indices for determining the number of clusters. The indices are based largely on Milligan and Cooper (1985), with the addition of the Dunn index (Dunn, 1974), Silhouette statistic (Rousseeuw, 1987), Gap statistic (Tibshirani, 2001), Dindex (Lebart, 2000), SD index and SDbw index (Halkidi et al., 2000, 2001), and Statistic of Hubert (Hubert and Arabie, 1985).

Its usage is:

> NbClust(data, diss = "NULL", distance = "euclidean", min.nc = 2, max.nc =15, method = "ward", index = "all", alphaBeale = 0.1)

The arguments are:

- data - matrix or dataset (the only mandatory argument)

- diss - dissimilarity matrix to be used. By default, diss="NULL", but if it is replaced by a dissimilarity matrix, distance should be "NULL".
- Distance - the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".
- min.nc - minimal number of clusters, between 1 and (number of objects - 1)
- max.nc - maximal number of clusters, between 2 and (number of objects - 1), greater or equal to min.nc. By default, max.nc=15.
- method - the cluster analysis method to be used. This should be one of: "ward", "single", "complete", "average", "mcquitty", "median", "centroid", "kmeans".
- Index - the index to be calculated. This should be one of : "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sdindex", "dindex", "sdbw", "all" (all indices except GAP, Gamma, Gplus and Tau), "alllong" (all indices with GAP, Gamma, Gplus and Tau included).
- alphaBeale - significance value for Beale's index.

Like always, you can type "**`?NbClust`**" at the **R** prompt for additional documentation.

Let's begin by typing:

```
NbClust(envdata,distance="euclidean",min.nc=2, max.nc=10, method="ward.D2")
```

Results returned include the amount of support for different number of clusters and the number of clusters based on the majority rule.

[All.index] contains the values of indices for each partition of the dataset obtained with a number of clusters between min.nc and max.nc.

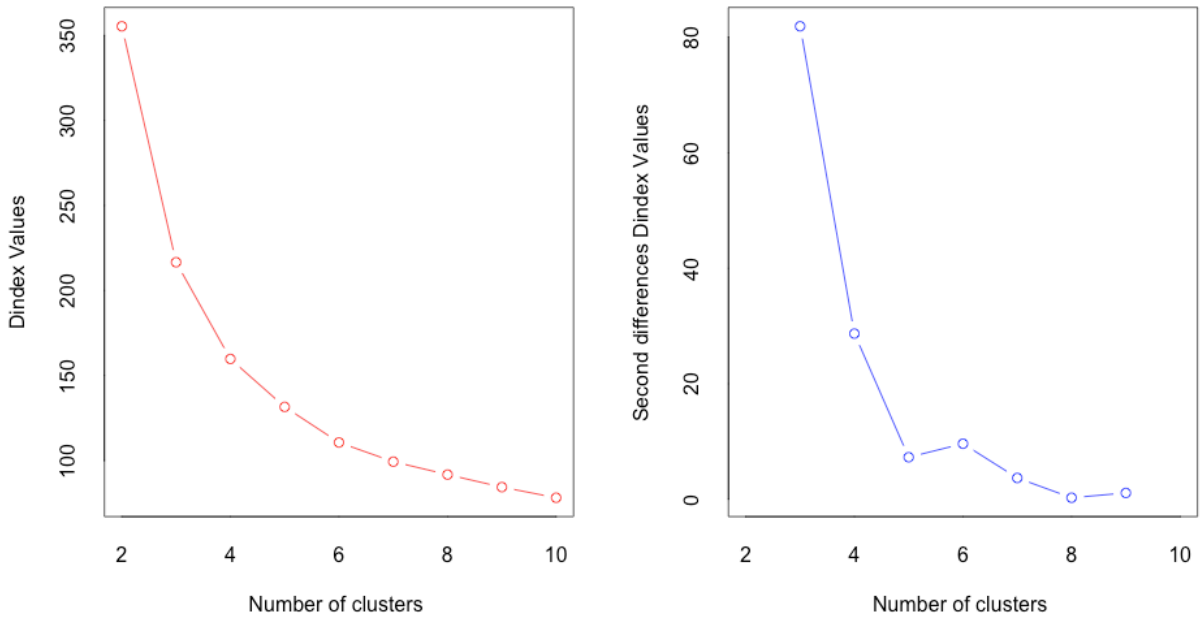
[All.CriticalValues] Critical values of some indices for each partition obtained with a number of clusters between min.nc and max.nc.

[Best.nc] Best number of clusters proposed by each index and the corresponding index value.

```
$Best.nc
                   KL          CH Hartigan       CCC          Scott
Number_clusters    3.0000      10.0000         3.0000       2.0000
Value_Index        3.7788      269.1251        43.6893      7.6569
…
```

You can use the majority rule to select the number of clusters or consider only the indices that have performed best in simulations studies (e.g., Milligan and Cooper 1985). Top-5 indices in Milligan and Cooper (1985) are: CH index, Duda index, Cindex, Gamma and Beale index. For binary data, Dimitriadou et al. (2002) found that Ratkowsky –Lance indiex performed the best.

Finally, the package alo creates plots according to the Hubert index and D Index.




In case you are thirsty for more, other functions exist to compute clustering indices! The function **`clustIndex()`** in the cclust library is a nice option.

## 6. Cluster Diagnostics: Comparing two clusters

There are times that you will be interested in comparing different cluster solutions that are based on different cluster algorithms (e.g., complete vs. average linkage) or datasets (e.g., species vs. environmental clustering) according to the same set of objects.

Many approaches exist, but here we will ultilize the clusteval library.

Let's use the cluster_similarity() function.

Its usage is:

```
cluster_similarity(labels1, labels2, similarity = c("jaccard", "rand"), method = "independence")
```

The arguments are:

- labels1 – a vector of n clustered labels
- labels2 – a vector of n clustered labels
- similarity – the similarity statistics to calculate
- method – the model under which the statistic was derived

Details:

To calculate the similarity, the 2x2 contingency table is computed, consisting of the following cells:

n_11 - the number of observation pairs where both observations are comembers in both clusterings

n_10 - the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 - the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 - the number of observation pairs where neither pair are comembers in either clustering

Let's begin by determining object membership according to three different clustering algorithms by typing:

```
sitecl.class.ave<-cutree(sitecl.ave,k=9)
sitecl.class.sin<-cutree(sitecl.sin,k=9)
sitecl.class.com<-cutree(sitecl.com,k=9)
```

Next, let's calculate the commonly used Rand Index (Rand 1971) to estimate the similarity between the cluster-derived memberships based on average linkage and single linkage.

```
cluster_similarity(sitecl.class.ave, sitecl.class.sin, similarity = "rand",
method = "independence")
```

This will return a value of 0.577 (range from 0 to 1), indicating moderate similarity.

Next, let's calculate the Rand Index to estimate the similarity between the cluster-derived memberships based on complete linkage and single linkage.

```
cluster_similarity(sitecl.class.com, sitecl.class.sin, similarity = "rand",
method = "independence")
```

This will return a value of 0.461, indicating lower similarity. Given what you know about the clustering algorithms, does this result make sense?

We can also look at the co-membership table for this last comparison (see n_11, n_10, n_01, n_00 above) by typing:

```
comembership_table(sitecl.class.sin,sitecl.class.com)
```

## 7. Cluster Diagnostics: Describing the clusters

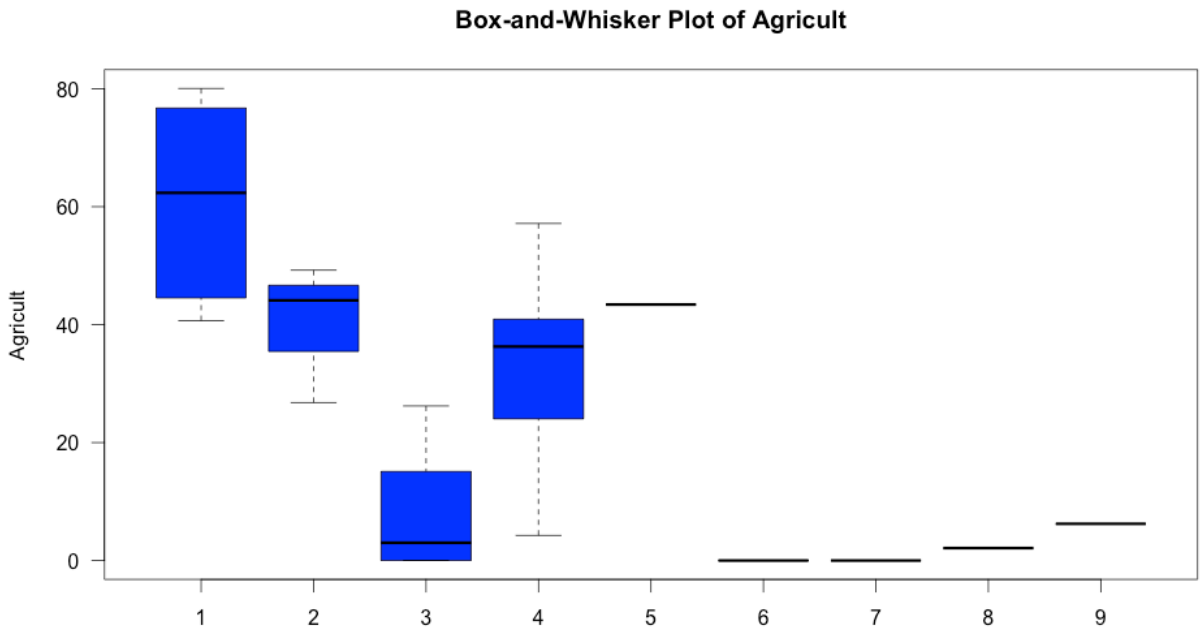
Once we have examined the clusters, we may want to describe the differences among clusters in ecological terms. In particular, we may wish to explore univariate differences among clusters based on the intrinsic variables used to cluster the samples or on extrinsic variables not used in the clustering. Begin by typing:

```
envdata.new<-cbind(sitecl.class,envdata)
```

This will bind the cluster classes (computed above) with the original dataset. You will notice that cluster membership in the 9 classes is now the first column of envdata.new matrix. Now we use box.plots()to produce separate box-and-whisker plots for each variable for each cluster, by typing:

```
box.plots(envdata.new,by='sitecl.class')
```

Here is the plot that is returned for agricultural landuse. What can you infer regarding differences between clusters?




---

**OTHER LIBRARIES FOR HIERARCHICAL AGGLOMERATIVE CLUSTERING**

The **`agnes()`** function (an abbreviation for agglomerative nesting) in the Cluster library provides an alternative to **`hclus().`** The only difference is that **`agnes()`** provides a novel graphical display and allows for conducting flexible beta clustering. Note that **`agnes()`** does not have an option for the median or centroid linkages. Feel free to explore the **`agnes()`** output by typing (don't forget to plot the results!).

**Flexible Beta Clustering**

In recent years many ecologists have expressed a preference for a hierarchical clustering algorithm call "flexible Beta" developed by Australian ecologists Lance and Williams (1966). Lance and Williams determined that because all hierarchical agglomerative algorithms operate similarly, but differ in the calculation of multi-member dissimilarity, that it was possible to generalize the algorithms to a single algorithm with four parameters, alpha_1, alpha_2, beta, and gamma. In Table 8.8 of Legendre and Legendre (1998) the parameter values to recreate many of the classical algorithms are presented. This table is reproduced below:

Values of parameters $\alpha_h$, $\alpha_i$, $\beta$, and $\gamma$ in Lance and Williams' general model for combinatorial agglomerative clustering. Modified from Sneath & Sokal (1973) and Jain & Dubes (1988).

| Clustering method | $\alpha_h$ | $\alpha_i$ | $\beta$ | $\gamma$ | Effect on space A |
|---|---|---|---|---|---|
| Single linkage | 1/2 | 1/2 | 0 | −1/2 | Contracting* |
| Complete linkage | 1/2 | 1/2 | 0 | 1/2 | Dilating* |
| UPGMA | $\frac{n_h}{n_h+n_i}$ | $\frac{n_i}{n_h+n_i}$ | 0 | 0 | Conserving* |
| WPGMA | 1/2 | 1/2 | 0 | 0 | Conserving |
| UPGMC | $\frac{n_h}{n_h+n_i}$ | $\frac{n_i}{n_h+n_i}$ | $\frac{-n_h n_i}{(n_h+n_i)^2}$ | 0 | Conserving |
| WPGMC | 1/2 | 1/2 | −1/4 | 0 | Conserving |
| Ward's | $\frac{n_h+n_g}{n_h+n_i+n_g}$ | $\frac{n_i+n_g}{n_h+n_i+n_g}$ | $\frac{-n_g}{n_h+n_i+n_g}$ | 0 | Conserving |
| Flexible | $\frac{1-\beta}{2}$ | $\frac{1-\beta}{2}$ | $-1 \le \beta < 1$ | 0 | Contracting if $\beta \approx 1$<br>Conserving if $\beta \approx -.25$<br>Dilating if $\beta \approx -1$ |

* Terms used by Sneath & Sokal (1973).

Of more interest here is a special case where alpha_1 = alpha_2 = 0.625, beta = -0.25, and gamma = 0. Observe that not only are alpha_1 and alpha_2 equal, it is also the case that beta = 1 - alpha_1 - alpha_2. This set of parameters returns a satisfactory result. To perform a flexible-beta using agnes(), the call is a little more complicated than we have seen for other clustering algorithms. We need to specify alpha_1, alpha_2, beta, and gamma in the call, using the par.method argument.

```
sitecl2.flb <- agnes(site.eucd, method='flexible',par.method=c(0.625,0.625,-0.25))
```

The defaults, however, are that alpha_1 = alpha_2, beta = 1 - alpha_1 - alpha_2, and gamma = 0. So we can simply specify alpha_1 and get the desired results.

The plot function for an object of class "agnes" has two panels. First, the "banner plot" is given, where the height of fusions is shown as a white bar on a red background. The second plot gives the more traditional dendrogram. Both plots include the agglomerative coefficient.

You can return the class memberships using the same approach implemented above. Type:

```
sitecl.class<-cutree(sitecl2.flb,k=9)
```

---

**CONDUCT HIERARCHICAL DIVISIVE CLUSTERING**

In contrast to agglomerative hierarchical clustering, there are only a limited number of divisive hierarchical algorithms that are available in R. The diana() algorithm constructs a hierarchy of clusterings, starting with one large cluster containing all n objects. Clusters are divided until each cluster contains only a single object. At each stage, the cluster with the largest diameter is selected

(the diameter of a cluster is the largest dissimilarity between any two of its objects). To divide the selected cluster, the algorithm first looks for its most disparate object (i.e., which has the largest average dissimilarity to the other objects of the selected cluster). This object initiates the "splinter group". In subsequent steps, the algorithm reassigns objects that are closer to the "splinter group" than to the "old group". The result is a division of the selected cluster into two new clusters. To compute a divisive hierarchical clustering using diana(), type the following:

```
sitecl3.dia<-diana(site.eucd)
```

---

**CONDUCT MONOTHETIC HIERARCHICAL DIVISIVE CLUSTERING**

If you are interested in conducting monothetic clustering (i.e., using a single descriptor as the basis for the partitioning of the data at each step), then I suggest you check out the mona() function in the Cluster library. This function returns a list representing a divisive hierarchical clustering of a dataset with binary variables only. Try typing the following:

```
sitecl4.mona<-mona(speocc)
```

---

**OPTIONAL READINGS**


Belbin, L., and C. McDonald. 1993. Comparing three classification strategies for use in ecology. Journal of Vegetation Science 4:341-348.

Gauch, H.G. and R.H. Whittaker. 1981. Hierarchical classification of community data. Journal of Ecology 69: 537-557.

Hunter, J.C. and R.A. McCoy. 2004. Applying randomization tests to cluster analysis

Jackson, D.A., Somers, K.M. and H.H. Harvey. 1992. Null models and fish communities: evidence of non-random patterns. Am. Nat. 139: 930-951.

Kaufman, L.A. and P.J. Rousseeuw. 1990. Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, New York, NY, US.

Milligan, G. W., and M. C. Cooper. 1985. An examination of procedures for determining the number of clusters in a data set. Psychometrika 50:159–179.

Strauss, R.E. 1982. Statistical significance of species clusters in association analysis. Ecology 63: 634-639.


---

**EXERCISE**

**Purpose**

In this exercise, we will explore the advantages and disadvantages of different clustering algorithms, and engage in the necessary steps of cluster analysis, including (1) computing an appropriate dissimilarity/distance matrix, (2) computing hierarchical clustering, (3) examining clustering results, (4) evaluating the cluster solution, (5) determining the number and stability of clusters, and (6) describing the clusters.

**Tasks**

1. Perform a cluster analysis on your own data (or the class data), using centroid, average, single, and complete linkage. Produce a dendrogram for each clustering method.
    a. Which clustering method performed best?
    b. For the best clustering method, how many clusters would you define?

2. Using a selected clustering outcome (from above), perform a statistical test on each variable testing the null hypothesis of equal cluster means.
3. Data standardization/transformation can play an important role in cluster analysis. Examine this yourself by comparing cluster results from raw vs. transformed data.
4. Flexible Beta clustering provides ultimate flexibility in terms of controlling the outcome of cluster analysis. Compare the dendrograms produced by using different beta values (holding alpha and gamma constant) with respect to space contracting, conserving and dilating.

**You will often be required to write a series of commands to help explore the power of multivariate statistics. Below is how you might answer question #4 above.**

```
# Create a vector with values from -1 to 1 by increments of 0.01
beta<-seq(-1,1,by=0.01)
size=length(beta)

# Create an empty vector to store the agglomerative coefficient values (note
# that we are defining the size equal to beta)
aggcoef<-vector(length=size)

# Create a loop to cluster the data using increasing values of beta and
# record the agglomerate coefficient for each cluster solution
for (j in 1:size) {
temp<-agnes(site.eucd, method='flexible',par.method=c((1- beta[j])/2,(1-
beta[j])/2,beta[j]))
aggcoef[j]<-temp$ac
}

plot(beta,aggcoef,xlab="Beta coef.",ylab="Agglomerate coef.")
```