5. Non-hierarchical Cluster Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Perform non-hierarchical divisive clustering

R Libraries: vegan, cluster, pvclust
R Source: biostats

BACKGROUND

Two general approaches to clustering are commonly used: hierarchical clustering (discussed in the previous chapter) and non-hierarchical partitioning. Non-hierarchical cluster analysis merely assigns each object or variable to a cluster in a multidimensional space without showing the interrelationships between objects. K-means is the most commonly used non-hierarchical clustering approach used in ecology. In k-means clustering, objects are assigned to k clusters (k being defined in advance), based on their nearest Euclidean distance to the mean of the clusters. The mean of the cluster is iteratively recalculated until no more assignments are made and cluster means fall below a predefined cut-off value or until the iteration limit is reached. Different means for each cluster are ideally obtained for each dimension used in the analysis, as indicated by high F-values from the respective analyses of variance. Unlike hierarchical clustering, k-means clustering does not require prior computation of dissimilarity matrix among objects and is therefore more adapted to large data sets (e.g. few thousand objects) where computing power is an issue. However, the method is quite sensitive to outliers, which are usually removed before performing the analyses (Legendre and Legendre 1998).

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 and cluster 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)
```

CONDUCT NON-HIERARCHICAL DIVISIVE CLUSTERING

Non-hierarchical divisive clustering is typically used when the objective is to summarize a very large data set into a much smaller set of groups, where the sole criterion for clustering samples is maximizing within-cluster similarity. Often, this is a precursor to subsequent analyses based on the newly formed clusters, and it may even involve non-hierarchical clustering to elucidate relationships among the newly formed clusters.

K-MEANS CLUSTER ANALYSIS IN R

Although most cluster functions in R operate on a dissimilarity/distance matrix, K-means is based directly on the object-by-descriptor dataset. Recall from your lecture that K-means clustering is based on the Euclidean distance among objects. However, we still must account for the situations where the descriptors (variables) are measured on different scales. In this example, we need to standardize the environment matrix (mean = 0, variance = 1) in order to control for this fact, by typing:

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

Now, let's calculate the Euclidean distance matrix, by typing:

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

Determine the number of clusters

In contrast to hierarchical clustering, with non-hierarchical clustering we typically need to specify the number of clusters sought. In some cases we will know the number of clusters we want to identify. However, the majority of the time we don't have a good idea of how many clusters to expect and we would like characteristics of the data to determine how many clusters to keep. In this case, we can construct a scree plot of some objective criterion against the number of clusters and look for indications of the "best" number of clusters to retain. One possible criterion is the average within-cluster dissimilarity, which declines monotonically as the number of clusters increases; it typically declines rapidly at first and then levels off somewhat. An abrupt leveling off or "elbow" in the scree plot provides an indication of the number of clusters to keep.

Another criterion is the average silhouette width. For each observation i, the silhouette width s(i) is defined as follows:

> Put a(i) = average dissimilarity between i and all other points of the cluster to which i belongs (if i is the only observation in its cluster, s(i) = 0 without further calculations). For all other clusters C, put d(i,C) = average dissimilarity of i to all observations of C. The smallest of these d(i,C) is b(i) = min_C d(i,C), and can be seen as the dissimilarity between i and its "neighbor" cluster, i.e., the nearest one to which it does not belong.
>
> Finally, s(i) = [b(i) - a(i)] / max[a(i), b(i)].

Observations with a large s(i) (almost 1) are very well clustered, a small s(i) (around 0) means that the observation lies between two clusters, and observations with a negative s(i) are probably placed in the wrong cluster. We can plot the average silhouette width for all observations against the number of clusters in a scree plot. We expect the average silhouette width to be highest when the number of clusters equals the natural clustering of the data. We can produce a scree plot of both objective criteria using the **`nhclus.scree()`** function in biostats, as follows:

```
nhclus.scree(site.eucd,max.k=44)
```

Note: the scree plot will depict the objective criteria for 2 to max.k number of clusters. Obviously, max.k cannot exceed the number of sample observations N and it probably makes no sense for max.k to approach N. The following plot should be produced:

From the scree plot you will notice that the average silhouette width is highest for 2 and 8 cluster solutions. You might also notice that there doesn't appear to be a clear elbow in the dissimilarity curve, therefore this provides little guidance for selecting the number of clusters in the data.

Compute K-means clustering

The K-means algorithm is a non-hierarchical method to cluster objects based on attributes into K partitions. In other words, given *n* objects in a *p*-dimensional space, this algorithm determines a partition of objects into K groups, or clusters, such that the objects within each cluster are more similar to each other than to objects in other clusters. The function **`kmeans`** (Stats library) is available to perform this cluster analysis. Several algorithms have been developed that accomplish the overall objective of identifying stable clusters and **`R`** gives you the option of specifying which algorithm to use. Nuances in each algorithm may result in slightly different cluster assignment patterns but, in general, the default algorithm setting "Hartigan-Wong" provides the most stable solutions.

The function's usage is:

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm)

The arguments are:

- **x** – object-by-descriptor matrix
- **centers** – the number of clusters desired.
- **inter.max** – maximum number of iterations the clustering algorithm should be employed (default is 10).
- **nstart** – number of random starts the algorithm should be initiated with to ensure that a stable clustering solution is reached (default is 1).
- **algorithm** – specify which clustering algorithm to use. Options include
    - Hartigan-Wong (default if algorithm is not specified)
    - Lloyd
    - Forgy
    - MacQueen

For additional documentation type "**?kmeans**" at the **R** prompt.

For this demonstration we'll use the transformed environmental dataset (see above). Now, let's perform k-means clustering. We'll request 8 clusters (based on the maximum average silhouette value from above – not including a cluster number of 2) and use a reasonably large number of iterations and random starts to ensure that a stable solution is converged upon. Begin by typing:

```
sitecl.kmeans<-kmeans(envdata.tra,centers=8,iter.max=10000,nstart=25)
```

Let's look at the results. Type:

```
sitecl.kmeans
```

Your results should look something like what appears below (why does it look different?)

```
K-means clustering with 8 clusters of sizes 5, 2, 6, 8, 7, 15, 1, 1

Cluster means:
    Sinuosity       Slope     WDRatio    SubEmbed     HabQual       Elev
1  0.34211414 -0.39076272  0.70826713 -0.70880097  1.26784743  0.4639215
2  0.78144578 -0.13148937 -0.22203311 -0.81992249  0.03486025 -1.4233593
3 -0.45547873  0.04137107  1.37458021  0.02519612 -0.16430947  0.9483609
…

      RoadDen   Agricult   HumanUse   BasinAre
1 -0.10433884  0.3062505  0.1452294  1.9760121
2  3.96706642 -0.9392717  1.3381329 -0.2896444
3 -0.46556915 -0.6329201 -0.6848619  0.7085082
…

Clustering vector:
 S1  S2  S3  S4  S5  S6  S7  S8  S9 S10 S11 S12 S13 S14 S15 S16 S17 S18
  5   5   4   6   6   1   4   4   3   6   6   7   2   8   4   2   5   1
S19 S20 S21 S22 S23 S24 S25 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36
  6   6   6   5   5   6   6   5   1   6   6   4   4   4   4   6   1   5
S37 S38 S39 S40 S41 S42 S43 S44 S45
  3   3   3   3   6   3   1   6   6

Within cluster sum of squares by cluster:
[1] 15.96343  9.23792 25.08302 29.99266 20.21755 46.68402  0.00000
[8]  0.00000

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

From the above output, "Cluster means" indicates the average value of the descriptor within each cluster; "Clustering vector" identifies the cluster identity of each site.

You can extract specific results from the newly created object **`sitecl.kmeans`** by referencing the available components (this is true of any object in R). For example, you can obtain the site classifications by typing:

```
sitecl.kmeans$cluster
```

Remember that you can also create a new table containing the extracted information by setting a new object name equal to the previous comment. For example, type:

```
sitecl.class<-sitecl.kmeans$cluster
```

Finally, also remember that you can examine the available components of any R object by typing:

```
names(sitecl.kmeans)
```

The analysis above yields a single partition with a pre-defined number of groups. You can try several solutions with different k values by using the function **cascadeKM()** which is a wrapper for **kmeans()**.

The function's usage is:

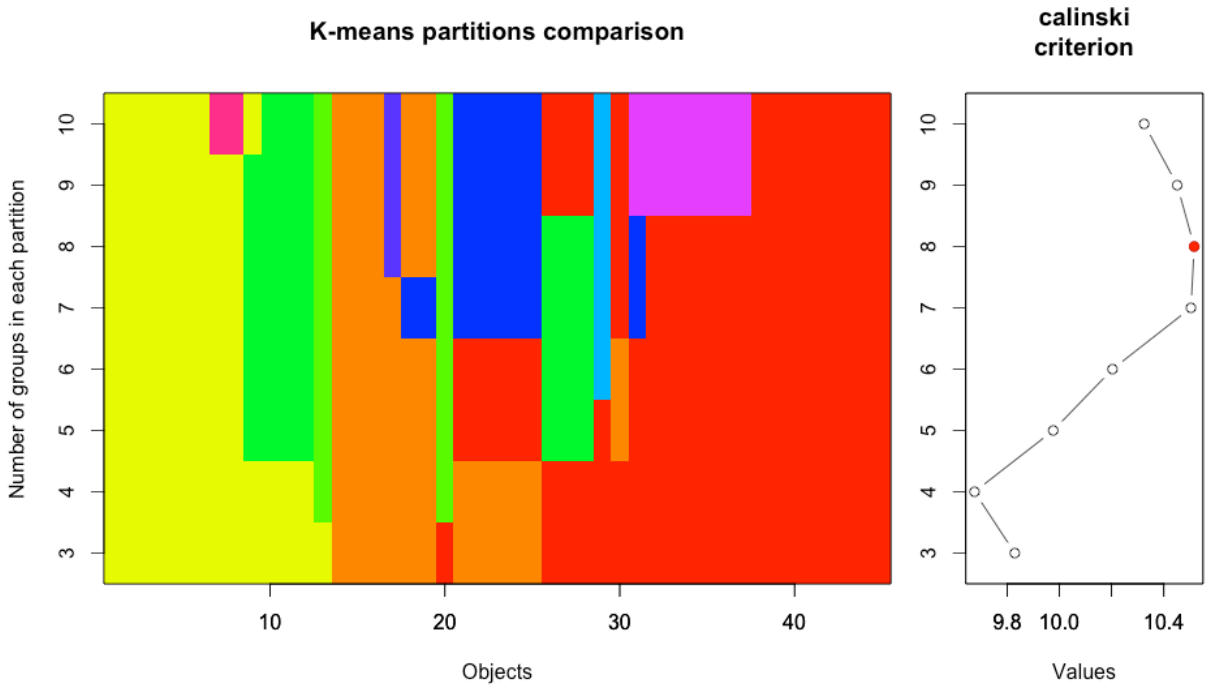
cascadeKM(data, inf.gr, sup.gr, iter = 100, criterion = "calinski")

The arguments are:

- Data - The data matrix. The objects (samples) are the rows.
- inf.gr - The number of groups for the partition with the smallest number of groups of the cascade (min).
- sup.gr - The number of groups for the partition with the largest number of groups of the cascade (max).
- iter - The number of random starting configurations for each value of K.
- criterion - The criterion that will be used to select the best partition. The default value is "calinski", which refers to the Calinski-Harabasz (1974) criterion. The simple structure index ("ssi") is also available. Other indices are available in function clustIndex (package cclust). In our experience, the two indices that work best and are most likely to return their maximum value at or near the optimal number of clusters are "calinski" and "ssi".

Let's begin by typing:

```
sitecl.kmeans.cas<-cascadeKM(envdata.tra,inf.gr=3,sup.gr=10,iter=100)
plot(sitecl.kmeans.cas,sortg=T)
```



The plot shows the group attributed to each object for each partition (rows of the graph). The groups are represented by different colors; there are two colors for k=2, three colors for k=3, etc … The graph to the right illustrates the values of the Calinski criterion for different values of the k. You can type:

```
sitecl.kmeans.cas$results
```

if you want the values that are contained in this plot. The Calinski index represents the F-statistic comparing the among-group to the within-group sum of squares of the partition. In other words, larger values provide more support for a particular number of clusters/groups. How many clusters does this analysis support?

Evaluating the number of clusters using randomization

Recall from Chapter 4 that it is possible to evaluate the "goodness-of-fit" of the selected number of clusters using a randomization test. To do so, one must typically write a short script (i.e., series of commands) in R. The idea of writing a script can be taunting, so let's take this one step at a time.

The randomization test will involve the following steps:

1. Define initial parameters
2. Perform K-means clustering on the original data and calculate the within-groups sum-of-squared error (SSE) for the number of cluster solutions selected by the user
3. For a number of different cluster numbers, randomize the order of the columns in the original matrix to create a random dataset
4. Perform K-means clustering on the random data and calculate the within-groups SSE
5. Repeat steps 3 and 4, and large number of times (here, I selected 999)
6. Compare the actual SSE to the distribution of random SEE

```
no.clus<-10
no.ran<-999
no.var<-10
```

Maximum number of clusters to consider
Number of randomizations to perform
Number of descriptors in the dataset

```
sse<-matrix(,nrow=no.ran+1,ncol=no.clus-1)
randata<-matrix(,nrow=nrow(envdata.tra),ncol=ncol(envdata.tra))
```

Creates an empty matrix to store within SSE values from the k-means on the original data (first row) and randomized data (rows 2, 3 … no.ran), where each column is a different number of clusters

Also creates an empty matrix to store the randomized dataset

```
for (j in 3:no.clus) {
```

For loop that repeats all commands between { and } a total of j times, where j = the number of clusters to consider starting with 3

```
sitecl.kmeans<-kmeans(envdata.tra,centers=j,iter.max=1000,nstart=5)
sse[1,j-1]<-sum(sitecl.kmeans$withinss)
```

Performing k-means using the original data, calculating the SSE and storing results in the first row and the j-1 column (e.g., column 1 for a two cluster solution)

```
for (k in 1:no.ran) {
```

For loop that repeats all commands between { and } a total of k times, where k = the number of randomizations

```
      for (c in 1:no.var) {
            randata[,c]<-envdata.tra[,c][sample(nrow(envdata.tra))]
      }
```

For loop that randomly selects values from each column (i.e., randomization)

```
      ran.kmeans<-kmeans(randata,centers=j,iter.max=1000,nstart=5)
      sse[k+1,j-1]<-sum(ran.kmeans$withinss)
```

Performing k-means using the random data, calculating the SSE and storing results in the k+1 row and j-1 column

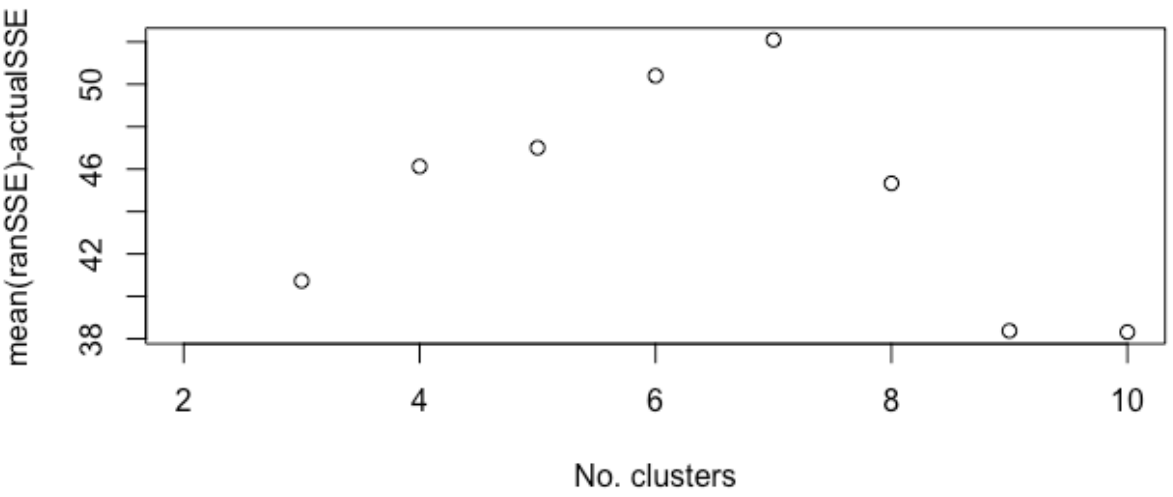
```
      }
}
```

Closing the cluster (j) and randomization (k) for loops

```
diff<-colMeans(sse)-sse[1,]
plot(seq(2,no.clus,by=1),diff,xlab="No. clusters",ylab="mean(ranSSE)-
actualSSE")
```

Calculating the difference between the average SSE based on the random k-means and the true SSE based on the actual data. Then the results are plotted.

Based on this information, what level of support do we have for the effectiveness of k-means based on different numbers of clusters? How many clusters would you pick for subsequent interpretation?



PARTITIONING AROUND MEDOIDS (PAM)

A number of other non-hierarchical divisive clustering algorithms exist, although these approaches are not commonly used in ecology. For examzple, the **`pam()`** or **`clara()`** functions in the Cluster library are methods for partitioning (clustering) of the data into k clusters 'around medoids', which is a more robust version of K-means clustering (Kaufman and Rousseeuw 1990). A particularly nice property is that PAM (Partitioning around medoids) allows clustering with respect to any specified distance metric (not just Euclidean distance, which k-means is based on). In addition, the medoids are robust representations of the cluster centers, which is particularly important in the common context that many elements do not belong well to any cluster. The **`clara()`** function is similar, but is designed for larger data sets with say >200 observations.

Like K-means, the number of clusters used in PAM is fixed in advance, and an initial set of cluster centers is required to start the algorithm.

The function's usage is:

pam(x, k, diss = inherits(x, "dist"), metric = "euclidean", medoids = NULL, etc …)

The arguments are:

- **x** – data matrix or data frame, or dissimilarity matrix or object, depending on the value of the diss argument.
- **k** – positive integer specifying the number of clusters
- **diss** – logical flag: if TRUE (default for dist or dissimilarity objects), then x will be considered as a dissimilarity matrix. If FALSE, then x will be considered as a matrix of observations by variables.
- **metric** – character string specifying the metric to be used for calculating dissimilarities between observations. The currently available options are "euclidean" and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If x is already a dissimilarity matrix, then this argument will be ignored.

For additional documentation type "**?pam**" at the **R** prompt.

We can compute a robust K-means clustering around 8 medoids (clusters) by typing:

```
sitecl.pam<-pam(site.eucd,k=8)
sitecl.pam
```

Your results should look somewhat close to what appears below.

```
Medoids:
     ID
[1,] "24" "S24"
[2,] "26" "S26"
[3,] "31" "S31"
[4,] "28" "S28"
[5,] "35" "S35"
[6,] "12" "S12"
[7,] "14" "S14"
[8,] "16" "S16"
Clustering vector:
 S1  S2  S3  S4  S5  S6  S7  S8  S9 S10 S11 S12 S13 S14 S15 S16 S17 S18
  1   2   3   4   1   5   3   3   5   4   4   6   1   7   3   8   2   5
S19 S20 S21 S22 S23 S24 S25 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36
  1   4   1   2   2   1   1   2   5   4   1   3   3   3   3   1   5   2
S37 S38 S39 S40 S41 S42 S43 S44 S45
  1   4   5   1   4   4   5   4   1
Objective function:
   build     swap
1.868102 1.859847

Available components:
[1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
[6] "clusinfo"   "silinfo"    "diss"       "call"
```

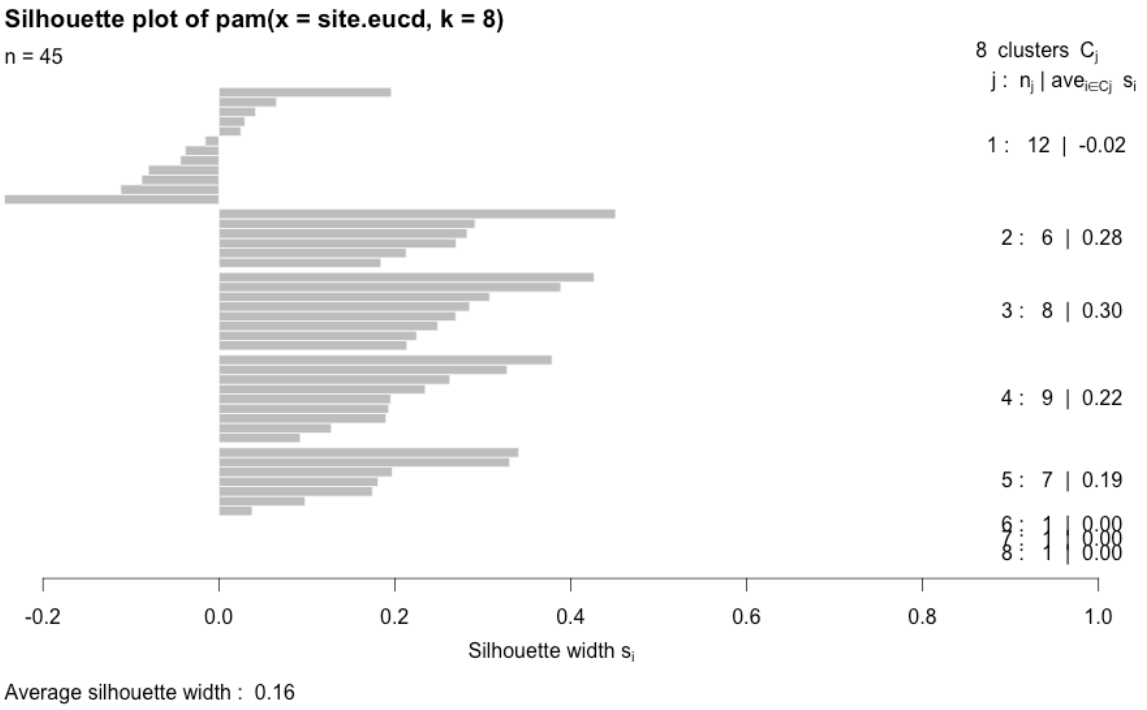
For large datasets, **pam()** may need too much memory or too much computation time. Then, **clara()** is preferable, see its documentation "**?clara**".

Once again you can extract specific results from the newly created object **sitecl.pam** by referencing the available components (this is true of any object in R). For example, you can obtain the site classifications by typing:

```
sitecl.pam$cluster
```

There is a special plotting function for outputs from the cluster library, including pam(). In R, just plot the pam object.

```
plot(sitecl.pam)
```



The plot is called a "Silhouette Plot", and shows for each cluster:

1. the number of objects per cluster = number of horizontal lines, also given in the right hand column
2. the means similarity of each object to its own cluster minus the mean similarity to the next most similar cluster (given by the length of the lines) with the mean in the right hand column
3. the average silhouette width

Objects which fit well within their cluster have a large positive Silhouette width; those which fit poorly have a small positive or even a negative Silhouette.

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.

Hartigan, J.A. and M.A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society, Series C 28: 100–108.

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

EXERCISE

Purpose

In this exercise, you will explore non-hierarchical approaches to cluster analysis.

Tasks

- Perform k-means clustering on your own data (or the class data) and analysis the results.
    - What type of cluster diagnostics should you perform?
    - How did you determine the "optimal" number of clusters?
- Perform a Partitioning around medoids using the class species abundance dataset and compare the results to a Partitioning around medoids based on the environmental data set.
    - How do these compare? That is, do sites that cluster according to similarities based on environment also cluster according to similarities based on species composition?