EBIO 5460

Exercise 5 – Indicator and Cluster Analyses

Indicator Species Analyses

Biologists often ask how species can act as indicators of different habitat types. Two of the most popular analyses available with which to ask this question are SIMPER and IndVal. SIMPER is the older of these methods and is the first demonstrated below. For the sake of consistency with earlier assignments, we will use the dune vegetation dataset once again.

```
library(vegan)

data(dune)
data(dune.env)
```

Let's relativize the dataset to focus on community composition:

```
dune.rel <- decostand(dune, "total")
```

SIMPER stands for 'similarity percentages,' and provides the contribution of each species to Bray-Curtis dissimilarities. The 'simper' function in R performs a pairwise comparison procedure among two or more groups. Here, we'll assess similarity percentages across the land management variables in the dune dataset.

```
sim <- simper(dune.rel, dune.env$Management)
summary(sim)
```

Of particular interest in the output are the average contribution of each species to overall dissimilarity, their standard deviations, and the cumulative contribution to the dissimilarities. The average abundances per group (A and B) are provided to interpret which group the species indicates.

Though SIMPER has been used widely, it is difficult to interpret. The difficulty arises in that the similarity percentages primarily reflect the most variable species both within and across groups, instead of distinctive species across groups. The best interpretation may be that, regardless of grouping, similarity percentages reflect each species' overall contribution to dissimilarity.

A newer method, called indicator value (IndVal) analysis, can better separate each species' predictive value across habitat types. For this analysis, we'll need the package 'indicspecies'. Please download and open the indicspecies package. Next, we can apply the 'multipatt' function from this package:

```
dune.ind = multipatt(dune.rel, dune.env$Management, func="IndVal.g", control =
how(nperm=999))
summary(dune.ind)
```

IndVal.g is based on the product of each species' frequency and relative abundance in each group, adjusted for sample size and square-root transformed. These values can be found in the 'stat' column of the summary output. Probability values are also provided, based on a permutational procedure. By default, the function only returns summaries for species with $P < 0.05$. We can easily see every species' maximum indicator value by adjusting the alpha level:

```
summary(dune.ind, alpha=1)
```

The multipatt function automatically calculates the indicator value for species within designated habitats and within all combinations of these habitats. However, we often want to focus the analysis on designated habitat types, which can be done by designating the argument 'duleg' to be TRUE:

```
dune.ind2 = multipatt(dune.rel, dune.env$Management, func="IndVal.g", duleg=TRUE, control = how(nperm=999))
summary(dune.ind2, alpha=1)
```

Related to questions of species' predictive power for different habitat types, ecologists also often ask whether species have preferences for various habitats. To address this type of question, a correlational approach may perform better than an indicator value approach because it can better detect negative preferences. Starting with species presence/absence data, we can use the multipatt function to calculate species habitat fidelities using Pearson's phi correlation adjusted for sample size:

```
dune.pa <- decostand(dune, "pa")
dune.r = multipatt(dune.pa, dune.env$Management, func = "r.g", duleg=TRUE, control = how(nperm=999))
summary(dune.r, alpha=1)
```

We can also calculate point biserial correlations, the counterpart to phi using abundance data. Again, we adjust for uneven sampling across habitats with ".g" in the function argument:

```
dune.r2 = multipatt(dune.rel, dune.env$Management, func = "r.g", duleg=TRUE, control = how(nperm=999))
summary(dune.r2, alpha=1)
```

Instead of species indicating different habitats, we may alternatively want to ask whether species are indicators of each other. This question can be asked using the 'indpower' function from vegan, which calculates the indicator power of species based on their co-occurrence:

```
indpower(dune.rel)
```

The above function returns a matrix of indicator power values with the indicator species as rows and the target species as columns.

Cluster Analyses

In some cases, we may not have data with habitats or groups assigned *a priori*. If this is the case, then a clustering approach can be used to determine groups. For cluster analyses, we'll first use the function 'agnes' from the package 'cluster'. This function determines agglomerative hierarchical clustering of sites based on dissimilarities, such as Bray-Curtis. It starts by considering each site as an individual cluster, then the algorithm merges similar sites until the entire dataset is included as one cluster. By default, the agnes function uses an averaging method, which measures the distance between two clusters as the average dissimilarity between sites in one group and those in another group.

```
library(cluster)

dune.bc<-vegdist(dune.rel,method="bray")
dune.cluster<-agnes(dune.bc, diss=TRUE, method="average", keep.diss=TRUE)
summary(dune.cluster)
pltree(dune.cluster)
```

Above, we clustered the dune vegetation data using Bray-Curtis dissimilarities and then produced a dendrogram to visualize the clustering. The height of the tree (ie. tree distance) represents the dissimilarity at each level of grouping, determined here by the averaging method. Looking at this tree from the top down, 3 distinct groups have an average dissimilarity near 0.7. We can cut the tree into these groups and ask whether there are indicator species for these clusters:

```
cut_tree<-cutree(dune.cluster,k=3)
ind_species<-multipatt(dune.rel,cut_tree,duleg=TRUE)
ind_species
summary(ind_species)
```

It may not always be obvious where to cut the tree to determine groups. To understand the best number of groups for the amount of dissimilarity explained, we can cut the tree at different points and compare mean indicator values. Here, we'll iterate the procedure to evaluate mean indicator values from data segregated into 2 to 15 groups:

```
mean_indcls <- numeric(15)
for (k_cuts in 2:15)
{
cut_tree<-cutree(dune.cluster,k=k_cuts)
ind_species<-multipatt(dune.rel,cut_tree,duleg=TRUE)
mean_indcls[k_cuts]<-mean(ind_species$sign$stat)
}
plot(mean_indcls)
```

From this graph, we can see that the largest increase in mean indicator species values occurs between 2 and 3 groups, and all additional groupings do not much improve these values. We may therefore conclude that 3 groups is best for explaining the indicator values.

There are several different methods that the agnes function can use to calculate distances between clusters. You may also see the averaging method above called UPGMA, or unweighted pair-group arithmetic averaging. Other methods include nearest neighbor clustering, based on the smallest dissimilarity between groups, and furthest neighbor clustering, based on the largest dissimilarity between groups.

```
#Nearest neighbor method
dune.cluster2 <- agnes(dune.bc, diss=TRUE, method="single", keep.diss=TRUE)
pltree(dune.cluster2)

#Furthest neighbor method
dune.cluster3 <- agnes(dune.bc, diss=TRUE, method="complete", keep.diss=TRUE)
pltree(dune.cluster3)
```

The Ward method is also commonly used and is based on minimizing variance in dissimilarities between groups:

```
dune.cluster4 <- agnes(dune.bc, diss=TRUE, method="ward", keep.diss=TRUE)
pltree(dune.cluster4)
```

We can also extract the estimated dissimilarity from a tree for all pairs of points, which is accomplished with the function 'cophenetic'. In the following, we extract the tree distances from the first cluster analysis we performed (UPGMA method), then we plot these distances against the original Bray-Curtis dissimilarities and assess the correlation between the two matrices. A correlation of 1 would mean that we have a perfect tree.

```
tree_dist<-cophenetic(dune.cluster)
plot(tree_dist, dune.bc)
cor(tree_dist, dune.bc)
```

Moreover, it is possible to display a cluster analysis in an ordination. Below, we display our original cluster analysis in an NMDS and a CCA.

```
dune.nmds = metaMDS(dune.rel, k=2)
plot(dune.nmds, display="sites")
ordicluster(dune.nmds, dune.cluster, col="blue")

dune.cca = cca(dune.rel)
plot(dune.cca, display="sites")
ordicluster(dune.cca, dune.cluster, col="blue")
```

In addition to agnes, there are many other clustering functions that can be used to ask different questions of the data. For instance, the 'hclust' function can be used to perform the same analyses as agnes. Another function, called 'kmeans', minimizes the sums of least squared Euclidean distances between groups. The kmeans function requires that you specify how many groups to compare, so we can use an iterative procedure to evaluate the mean SS for different numbers of groups. Instead of using a distance matrix for this function, we use the original data:

```
dune.k_means <- kmeans(dune.rel, 2)
dune.k_means

mean_cluster_ss <- numeric(15)
for (k in 2:16) {
k_means_cluster<-kmeans(scale(dune.rel),k)
mean_cluster_ss[k]<-sum(k_means_cluster$withinss/k_means_cluster$size)
}
mean_cluster_ss
```

In the output for mean SS, notice that a clustering of 2 may be best because there are higher SS with greater numbers of groups.

A more robust version of kmeans is 'pam', or positioning around medoids. This function minimizes the sum of dissimilarities instead of the sum of squared Euclidean distances. For this, we can specify the dissimilarity matrix for the cluster analysis:

```
cluster_med<-pam(dune.bc, 2, diss=TRUE, keep.diss=TRUE)
summary(cluster_med)
plot(cluster_med)
```

Like with kmeans, we must also specify the number of groups with the pam function, and we can use an iterative approach to decide how many groups are optimal. Below, we can see that 3 groups may be optimal for this dataset (similar to results using agnes) because it minimizes the summed dissimilarities, measured by the so-called 'silhouette' values:

```
mean_silhouette <- numeric(15)
for (k in 2:16) {
cluster_med_variable_k<-pam(dune.bc, k, diss=TRUE)
mean_silhouette[k]<- cluster_med_variable_k$silinfo$avg.width
}
mean_silhouette

cluster_med<-pam(dune.bc, 3, diss=TRUE, keep.diss=TRUE)
summary(cluster_med)
plot(cluster_med)
```

Application

We now have a full toolset with which to analyze multivariate datasets, and it is time to apply these tools to your own data. For class next week, please bring a 1 page summary of a complete multivariate analysis, including 1) a hypothesis test, 2) an ordination, and 3) an indicator species and/or cluster analysis. For instance, you might first apply a PERMANOVA, then visualize these results with NMDS, and finally conduct an indicator value analysis to identify species that are important for distinguishing different experimental groups. If you do not have a dataset, then feel free to use the Carrizo vegetation data included with Exercise 2.

Katie and I will collect your analysis summaries during class and return them to you with some feedback. These will not be graded, but will help you to prepare for presenting your analysis at the end of the semester.