EBIO 5460
Ordination Module 1, NMDS and PCoA
Analysis Exercise

Non-metric Multidimensional Scaling. The NMDS procedure is iterative and takes place over several steps. Function metaNMDS is a wrapper function that combines several functions into one command. To begin, NMDS requires a distance matrix, or a matrix of dissimilarities.

Okay, so lets do it.

```
#To run the NMDS, we will use the function `metaMDS` from the vegan package
library(vegan)
# metaMDS requires a community-by-species matrix, here we will use the same dataset as last time.
data(dune)
data(dune.env)
dune.rel<-decostand(dune,"total")
```

The function `metaMDS` will take care of most of the distance calculations, iterative fitting, etc, We need simply to supply our sample unit by species matrix (dune.rel) and number of reduced dimensions (k=#).

```
dune.nmds=metaMDS(dune.rel, k=2)
```

And then we can look at the NMDS object

```
dune.nmds
```

This gives a quick overview of what metaMDS just did:

1. Transformation. The function will perform a Wisconsin double standardization (**wisconsin**) if data maximum values are >9, and if the values are very large (>50), a square-root transformation (**sqrt**). If you want to have full control over data transformations (you should, and particularly should if you have data types other than typical community data) you can set **autotransformation=FALSE**. In this case we just ran, the maximum data values did not trigger transformations but if would be listed here it they did.

2. Calculate a dissimilarity matrix. The default is Bray-Curtis dissimilarity, and because we did not specify, that is what was used here. Any other dissimilarity index in **vegdist** can be used using the argument distance. Consider that it is only the rank order of the dissimilarity metric that matters for NMDS.

3. Treatment of tied dissimilarity values. The default NMDS engine is **monoMDS**, which breaks tied values at maximum dissimilarity (with no shared species) using monotone regression to minimize stress. If you use **isoMDS** (probably not, see 4 below, be careful about handling ties).

4. NMDS with random starts. NMDS gets easily trapped into local optima, so it is important to start NMDS several times from random starts to be confident that you have found the global solution. **metaMDS** starts NMDS from several random starts (minimum number is given by try and maximum number by trymax). If a solution is better (has lower stress) than the previous best, it is taken as the new standard. Procrustes analysis compares two solutions, and if they are similar (small residuals) they are considered convergent (indicating a global solution). If you want to be

more certain of reaching a global solution, you can compare results from several independent runs.

There is an older alternative NMDS engine (**isoMDS** in the **MASS** package), but it does not have several random starts; for this reason, **monoMDS** (default in metaMDS) is preferable.

5. Then the results are scaled using postMDS, default options use centring, PC rotation and half-change scaling. Centring moves the origin to the average of the axes. PC rotation rotates the configuration so that the variance of points is maximized on the first dimension. With MDSrotate you can rotate the configuration so that the first axis is parallel to an environmental variable. Half-change scaling scales the configuration so that one unit means a decrease in half of the community similarity from perfect similarity.

6. The last step in metaMDS is that species scores are added to the final solution. Remember the dissimilarity matrix is based on similarity across sample units. Species scores are weighted averages using function **wascores**. These are helpful to include in visual representations of the NMDS.

You should see each iteration of the NMDS until a solution is reached (i.e., stress was minimized after some number of reconfigurations of the points in 2 dimensions). You can increase the number of default runs (each at a random starting point) using the argument "trymax=##"

```
dune.nmds=metaMDS(dune.rel, k=2, trymax=100)
```

Results also lists stress. Stress is calculated by regressing distances in an initial 2-D configuration against the observed (measured) distances, then determining the disagreement between 2-D configuration and predicted values from regression. If the 2-D configuration perfectly preserves the original rank orders, then a plot of one against the other should be a monotonic increase. The extent to which the points on the 2-D configuration differ from this monotonically increasing line determines the degree of stress (this is a Shepard plot, which we will do next). A next iteration repositions the points in 2 (or more) dimensions in the direction of decreasing stress, and repeat until stress is below some threshold.

Generally, stress < 0.05 provides an excellent representation in reduced dimensions, < 0.1 is great, < 0.2 is good, and stress > 0.3 provides a poor representation. You always should report the stress of a NMDS, as it is important for readers to interpret your analysis.

Let's examine a Shepard plot, which shows scatter around the regression between the interpoint distances in the final configuration (distances between each pair of communities) against their original dissimilarities. The stress of this analysis indicates a pretty good representation and the Shepard plot confirms.

```
stressplot(dune.nmds)
```

Now we can plot the NMDS

```
plot(dune.nmds, type="t")
```

Remembering the last exercise, where we compared management treatments using a PERMANOVA, let's use ellipses for differences in Management, based on standard error.

```
ordiplot(dune.nmds, type="n")
```

```
with (dune.env, ordiellipse(dune.nmds, Management, kind="se", conf=0.95, col="blue", lwd=2,
label=TRUE))
orditorp (dune.nmds, display="species", col="black", air=0.01)
```

Another visualization is to show convex hulls connecting the vertices of the points made by these treatments on the plot.

```
ordiplot(dune.nmds, type="n")
with (dune.env, ordihull(dune.nmds, Management, col="blue", lwd=2, label=TRUE))
orditorp (dune.nmds, display="species", col="black", air=0.01)
```

If you plot points (type=”p” in ordiplot), you can confirm that the ordihull function connects vertices of the points made for each treatment. Another visual variation is ordispider, which plots the centroid of each group in ordination space.

Lastly, let’s consider a continuous variable. In this case, map contour lines onto the plot using the function ordisurf:

```
ordiplot(dune.nmds,type="n")
with (dune.env, ordisurf(dune.nmds, A1, add = TRUE))
orditorp(dune.nmds,display="species",col="grey30",air=0.01)
```

Using **ordisurf,** you might notice that this does more than graphically represent contour lines, it also includes a **gam** function (REML is the restricted maximum likelihood estimate). There are also ways to fit environmental factors to NMDS results using **envfit**. These give P-values and are based on permutations, but consider them as post-hoc tests because they fit environmental (or other explanatory) vectors onto an ordination rather than starting from the data matrix (e.g., PERMANOVA). They also can consider categorical groups (factors) such as Management. We suggest considering these confirmatory but relying on hypothesis testing based directly on resemblance matrices (like PERMANOVA). If you do both, and get conflicting results, it might suggest that some default steps in the NMDS might be worth looking at.

```
data(dune.env)
ef<-envfit(dune.nmds, dune.env, permu=999)
ef
```

Principal Coordinates Analysis (PCoA). NMDS is best when your goal is to preserve the ordering relationships among objects, particularly into two or three dimensions. However, it does not preserve the exact distance relationships among objects and is computer intensive; if these considerations are important, you might use a PCoA.

PCoA is also called metric multidimensional scaling – you start from a dissimilarity matrix, and points are plotted in a way to make the distance between every pair of points as nearly as possible equal to their dissimilarity. This is a bit of a head spinner, but it is a way to obtain a Euclidean representation (Cartesian co-ordinate space) of a set of objects whose relationships are measured by non-Euclidean measures of resemblance. It should be reserved for cases where no Euclidean measure is appropriate (e.g., lots of shared zeros) and you prefer to use a similarity measure.

Let’s compute a matrix of Bray-Curtis similarities among sites, and subject this matrix to PCoA.
If the metric is non-euclidean (as in our case), then the PCoA may produce several negative eigenvalues in addition to the positive ones. In most applications, this does not affect the representation of the first

several axes. You will still receive a warning message, though, when this occurs. You also will receive a warning about species scores not being available; there is a way to project weighted averages of species abundances on a PCoA plot using the function **wascores**, we will do that too.

```
dune.bray<-vegdist(dune.rel)
dune.b.pcoa<-cmdscale(dune.bray, k=(nrow(dune)-1), eig=TRUE)
```

Check to see if negative eigenvalues affect the interpretation of the first several axes

```
dune.b.pcoa
```

We will discuss eigenvalues more next week as we move further into eigenvalue-based ordination methods; for now just know that each ordination axis has an eigenvalue, it is a measure of the strength of an axis, the amount of variation explained by the axis. We typically consider the first several axes, as they have greater eigenvalues and thus explain much of the variation in a dataset. Also note that if you redo this PCoA analysis with Euclidean distance, the eigenvalues/(n-1) should be eigenvalues of a PCA.

Now lets graph it.

```
ordiplot(scores(dune.b.pcoa)[,c(1,2)], type="t")
abline(h=0, lty=3)
abline(v=0, lty=3)
dune.wa <-wascores(dune.b.pcoa$points[,1:2], dune)
text(dune.wa, rownames(dune.wa), cex=0.7, col="red")
```

You can also add convex hulls or ellipses, using the same functions as you did with the NMDS.

Looking ahead. We have hinted at eigenvalues and eigenvalue-based ordinations, and that is where we will head next. These include Principle Component Analysis (PCA). The main difference is that the first step in a PCA is that it is plotted in a *s*-dimensional coordinate frame rather than based on a dissimilarity matrix. You can argue that PCA is often superior to PCoA because the PCA places points where they should be in multivariate space, but that depends on whether you trust Euclidean distances to describe your dataset.

Next steps to do for class. On one page, include the following:

a. Briefly give the general design of the multivariate dataset you are using.
b. State one hypothesis.
c. Test that hypothesis. State your results and the test used.
d. Depict your results using a NMDS (or a PCoA)

***If you are willing to share and get feedback, bring four copies to class next week ***