PCO and NMDS

Principal Coordinates Analysis (PCO)
- Introduction
- PCO vs. PCA
- PCO vs. NMDS
- PCO Computation
- PCO Example via MVSP

Nonmetric Multidimensional Scaling (NMDS)
- 8 Step determination
- NMDS Example via NT-SYS
- Mantel test for matrix comparison

Principal Coordinate Analysis (PCO)

In a previous lecture, we discussed a variety of measures of *resemblance* that are especially useful for biological data where many zeroes occur or only presence/absence data are available.

In such cases, the Euclidean distance (among objects) and the corresponding measures of covariance or correlation (among descriptors) do not provide acceptable models.

Gower (1966) described a method to obtain a Euclidian representation (Cartesian co-ordinate space) of a set of objects whose relationships are measured by any similarity or distance coefficient chosen by the user.

Principal Coordinate Analysis (PCO)

Gower's method became known as *Principal Coordinate Analysis* (PCO) or *metric multidimensional scaling* (MDS) or classical scaling. Note that MDS should not be confused with the *nonmetric* version (NMDS) that we will cover later.

The goal of PCO is to permit the positioning of objects in a space of reduced dimensionality while preserving their distance relationships as well as possible.

The value of PCO is that it permits the use of all types of variables, provided that a coefficient of appropriate type has been used to compute the resemblance hemi-matrix.

Metric vs. Non-metric Matrices

If the distance matrix is metric (i.e., no violation of the triangle inequality), the relationships among objects can, in most cases, be fully represented in Euclidean space.

If the distance matrix is non-metric (i.e., violation of the triangle inequality), the relationships among objects can often produce *negative eigenvalues*.

In most cases this does not impair the quality of the Euclidean representation obtained for the first few principle coordinates. Alternatively, the resemblance matrix can be transformed (e.g., square root or add a constant to distances D_{hi}) to avoid the problem altogether.

PCO vs. PCA

Thus, PCO differs from PCA in the way in which the the data swarm is constructed to begin with.

In PCO, the points are NOT plotted in an *s*-dimensional coordinate frame.

Instead, dissimilarities are calculated between every possible pair of objects, and the points plotted in such a way as to make the distance between every pair of points as nearly as possible equal to their dissimilarity.

PCO vs. PCA

One could argue that PCO is necessarily inferior to PCA because in PCA each point is placed exactly where it ought to be, whereas in PCO each point is only approximated based on a best-fit model of the dissimilarities.

Provided the approximation is close, the imprecision of PCO is of no *practical* consequence to the biologist attempting to understand order.

Unfortunately, there has been precious little research to explore what dissimilarity measures perform best with PCO under specific conditions (Podani & Miklos 2002).

PCO vs. NMDS

The method of NMDS also obtains ordinations of objects from any resemblance matrix. It is better than PCO in compressing the distance relationships among objects into *two or three dimensions*.

By construction, NMDS always obtains a Euclidean representation. PCO can only represent that portion of the matrix which actually *is* Euclidean (even when non-Euclidean distances are present). Comparatively speaking, NMDS is very computer intensive WRT PCO. However, present processor speeds do not limit such calculations anymore.

PCO Computation

Gower (1966) explained how to compute the principal coordinates of a resemblance matrix:

(1) The initial matrix needs to be a distance matrix **D** = [D_{hi}] (but may be an **S** matrix if you transform it first to a **D** matrix).

(2) Matrix **D** is transformed into a new matrix **A** by defining:

$$a_{hi} = -\frac{1}{2}D_{hi}^2$$

(3) Matrix **A** is centered to give matrix $\mathbf{\Delta}_1$ = [δ_{hi}], using eq.:

$$\delta_{hi} = a_{hi} - \overline{a}_h - \overline{a}_i + \overline{a}$$

Where the $\overline{a}_h$ and $\overline{a}_i$ are the row and column means of matrix **A**, respectively; and $\overline{a}$ is the mean of all a_{hi}'s in the **A** matrix.

PCO Computation

Note: in the particular case of distances computed using the Euclidean distance coefficient (D_1), it is possible to obtain the Gower-centered matrix $\mathbf{\Delta}_1$ directly without computing the above two equations.

(4) The eigenvalues and eigenvectors are computed and the latter are scaled to lengths equal to the square roots of the respective eigenvalues:

$$\sqrt{u'_k\, u_k} = \sqrt{\lambda_k}$$

Due to the centering, matrix $\mathbf{\Delta}_1$ always has at least one zero eigenvalue (may have more if **D** degenerates).

(5) After scaling, if the eigenvectors are written as columns, the rows of the resulting table are the coordinates of the objects in PCO space.

PCO Numerical Example

Let's do a numerical example and compute a PCO using a matrix of Euclidean distances (D_1). Confirm for yourself that this produces the identical result as a PCA, with the exception of the variable loadings.

NB: Information about the original variables is not passed on to the PCO algorithm; since PCO is computed from a distance matrix it can not give back the loadings of the variables. A method for computing them *a posteriori* will be described later.

PCO Numerical Example

Begin with a matrix of Euclidean distances **D** derived from 5 objects in data matrix **Y**:

$$\mathbf{D} = \begin{bmatrix} 0.00000 & 3.16228 & 3.16228 & 7.07107 & 7.07107 \\ 3.16228 & 0.00000 & 4.47214 & 4.47214 & 6.32456 \\ 3.16228 & 4.47214 & 0.00000 & 6.32456 & 4.47214 \\ 7.07107 & 4.47214 & 6.32456 & 0.00000 & 4.47214 \\ 7.07107 & 6.32456 & 4.47214 & 4.47214 & 0.00000 \end{bmatrix}$$

PCO Numerical Example

Derive matrix $\boldsymbol{\Delta}_1$ via Gower's centering equations:

$$\Delta_1 = \begin{bmatrix} 12.8 & 4.8 & 4.8 & -11.2 & -11.2 \\ 4.8 & 6.8 & -3.2 & 0.8 & -9.2 \\ 4.8 & -3.2 & 6.8 & -9.2 & 0.8 \\ -11.2 & 0.8 & -9.2 & 14.8 & 4.8 \\ -11.2 & -9.2 & 0.8 & 4.8 & 14.8 \end{bmatrix}$$

Note: The trace of this matrix is 56. This is (n - 1) = 4 times the trace of the covariance matrix computed from a PCA which is 14.

PCO

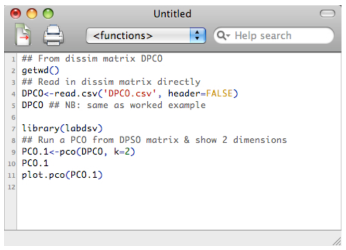
Numerical Example

Note that PCO eigenvalues / (n-1) = eigenvalues of PCA, *when using Euclidean distance.*

	Eigenvalues	
	λ_1	λ_1
Objects	Eigenvectors	
$\mathbf{x}_1$	-3.578	0.000
$\mathbf{x}_2$	-1.342	-2.236
$\mathbf{x}_3$	-1.342	2.236
$\mathbf{x}_4$	3.130	-2.236
$\mathbf{x}_5$	3.130	2.236
Eigenvalues of PCO	36.000	20.000
Eigenvalues of PCA	9.000	5.000
Length	√36=6	√20=4.472

PCO

Numerical Example via R



```
## From dissim matrix DPCO
getwd()
## Read in dissim matrix directly
DPCO<-read.csv('DPCO.csv', header=FALSE)
DPCO ## NB: same as worked example

library(labdsv)
## Run a PCO from DPSO matrix & show 2 dimensions
PCO.1<-pco(DPCO, k=2)
PCO.1
plot.pco(PCO.1)
```

PCO

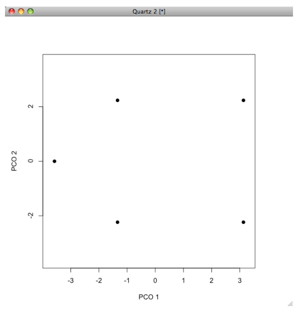
Numerical Example via R

```
> DPCO<-read.csv('DPCO.csv', header=FALSE)
> DPCO ## NB: same as worked example
    V1   V2   V3   V4   V5
1 0.00 3.16 3.16 7.07 7.07
2 3.16 0.00 4.47 4.47 6.32
3 3.16 4.47 0.00 6.32 4.47
4 7.07 4.47 6.32 0.00 4.47
5 7.07 6.32 4.47 4.47 0.00
> library(labdsv)
> ## Run a PCO from DPSO matrix & show 2 dimensions
> PCO.1<-pco(DPCO, k=2)
> PCO.1
$points
          [,1]      [,2]
[1,] -3.578313  0.000000
[2,] -1.340221 -2.234457
[3,] -1.340221  2.234457
[4,]  3.129377 -2.234457
[5,]  3.129377  2.234457

$eig
[1] 35.98271 19.97120

$x
NULL

$ac
[1] 0

$GOF
[1] 0.9998267 0.9998267

attr(,"class")
[1] "pco"
```

NB: Exact same results as when worked by hand!

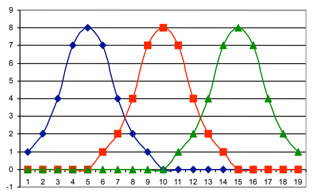
PCO

Numerical Example via R

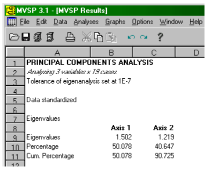


PCA vs. PCO

Let's construct a hypothetical data set of three unimodally distributed species arranged along an environmental gradient:



Site:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Species-1:	1	2	4	7	8	7	4	2	1	0	0	0	0	0	0	0	0	0	0
Species-2:	0	0	0	0	0	1	2	4	7	8	7	4	2	1	0	0	0	0	0
Species-3:	0	0	0	0	0	0	0	0	0	0	1	2	4	7	8	7	4	2	1

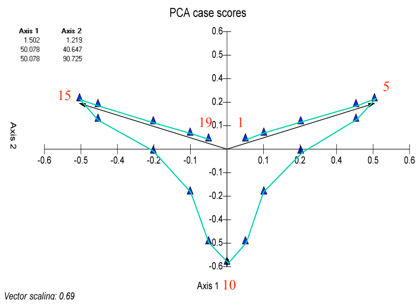


MVSP 3.1 - [MVSP Results]
File Edit Data Analyses Graphs Options Window Help

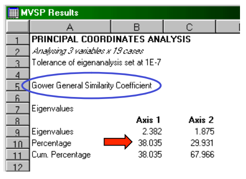
	A	B	C	D
1	**PRINCIPAL COMPONENTS ANALYSIS**			
2	*Analysing 3 variables x 19 cases*			
3	Tolerance of eigenanalysis set at 1E-7			
4				
5	Data standardized			
6				
7	Eigenvalues			
8		**Axis 1**	**Axis 2**	
9	Eigenvalues	1.502	1.219	
10	Percentage	50.078	40.647	
11	Cum. Percentage	50.078	90.725	

PCA Biplot

19 samples and 3 species:

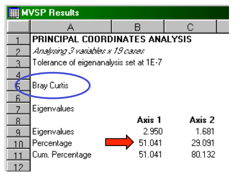


PCO w/ Gower & BC



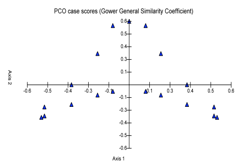
MVSP Results

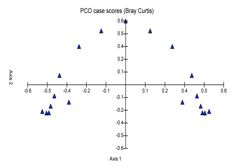
	A	B	C
1	**PRINCIPAL COORDINATES ANALYSIS**		
2	*Analysing 3 variables x 19 cases*		
3	Tolerance of eigenanalysis set at 1E-7		
4			
5	Gower General Similarity Coefficient		
6			
7	Eigenvalues		
8		**Axis 1**	**Axis 2**
9	Eigenvalues	2.382	1.875
10	Percentage	38.035	29.931
11	Cum. Percentage	38.035	67.966
12			



MVSP Results

	A	B	C
1	**PRINCIPAL COORDINATES ANALYSIS**		
2	*Analysing 3 variables x 19 cases*		
3	Tolerance of eigenanalysis set at 1E-7		
4			
5	Bray Curtis		
6			
7	Eigenvalues		
8		**Axis 1**	**Axis 2**
9	Eigenvalues	2.950	1.681
10	Percentage	51.041	29.091
11	Cum. Percentage	51.041	80.132
12			





PCA vs. PCO

The primary data matrix contains 50% zeroes. This leads us to be cautious WRT Euclidean distance and PCA. Indeed, an extreme horseshoe manifests itself (green line).

PCO using Gower's coefficient produces a very similar result. This speaks to the integrity of Gower's coefficient to "act like" Euclidean distance, but results in decreased resolution: note decrease in variance accounted for.

PCO using the Bray Curtis coefficient completely compresses the two terminal ends and results in poor resolution. But, note similar variance to PCA w/ED (better than PCO w/Gower).

Nonmetric Multidimensional Scaling (NMDS)

All reduced-space ordination methods start from an ordination of the objects in full-dimensional space and attempt to represent them in a few dimensions, while preserving as best as possible, the distance relationships among objects.

There are cases where the exact preservation of distances is not of primary importance, the priority being a representation in only 2- or 3-dimensions. In such cases, the goal is to plot similar objects together and dissimilar ones further apart. This is called the *preservation of ordering relationships among objects*. NMDS is used to accomplish this.

Nonmetric Multidimensional Scaling (NMDS)

Programs for NMDS were originally distributed only by Bell laboratories in NJ where it originated, but is now available in SAS, SPSS, PC-ORD and a variety of other MV stat packages.

Like PCO, NMDS is not limited to Euclidean distance matrices; it can produce ordinations of objects from any distance matrix.

Contrary to eigenvector methods such as PCA or PCO, NMDS calculations do not maximize the variability associated with individual axes of the ordination; NMDS axes are arbitrary, so the final plots can be rotated, centered, and inverted.

Nonmetric Multidimensional Scaling (NMDS)

Consider a distance matrix $\mathbf{D}_{n \times n} = [D_{hi}]$ computed using a measure appropriate to the data at hand.

This method is more *process-oriented* and proceeds by 8 steps. We will consider each in turn to better understand the process.

NMDS: Step-1

Specify the number *m* of dimensions chosen *a priori* for scaling the objects.

The output will provide coordinates of the *n* objects on *m* axes.

NMDS: Step-2

Construct an initial configuration of the objects in *m* dimensions, to be used as a starting point for the iterative adjustment process of steps 3 to 7. The way this initial configuration is chosen is critical because the solution on which the algorithm eventually converges depends to some extent on the initial positions of the objects (recall *local minimum* problem discussed previously).

The most common solutions to this initial starting conditions problem are:
(1) Run the program several times starting in different places,
(2) Initiate the run from an ordination such as PCO,
(3) If spatially structured, start from a spatial coordinate matrix,
(4) Start with 5 dimensions and work your way down.

NMDS: Step-3

Calculate a matrix of fitted distances d_{hi} in the ordination space, using one of Minkowski's metrics (D_6).

Most often one chooses the second degree of the Minkowski metric, which is the Euclidean distance.

In the first iteration, distances d_{hi} are computed from the initial (often random) configuration. In subsequent iterations, the configuration is that obtained in step 6.

NMDS: Step-4

Consider the use of a Shepard diagram comparing the fitted distances d_{hi} to the empirical (original) distances D_{hi}. Regress d_{hi} on D_{hi}. Values forecasted by the regression line are called: $\hat{d}_{hi}$

The type of regression can be linear, polynomial, or nonparametric (monotone). It is at the discretion of the user.

How the computer application handles ties is important (there are several methods for doing this) and may result in dramatic differences.

NMDS: Step-5a

Measure the goodness-of-fit of the regression using an objective function.

All of the objective functions used in NMDS are based on the sum of the squared differences between fitted values d_{hi} and the corresponding $\hat{d}_{hi}$ forecasted by the regression function; this is the usual sum of squared residuals of regression analysis.

Several variants have been proposed that are being used in NMDS software:

NMDS: Step-5b

$$\textit{Stress}\text{ (formula 1)} = \sqrt{\sum_{h,i}(d_{hi}-\hat{d}_{hi})^2 \Big/ \sum_{h,i} d_{hi}^2}$$

$$\textit{Stress}\text{ (formula 2)} = \sqrt{\sum_{h,i}(d_{hi}-\hat{d}_{hi})^2 \Big/ \sum_{h,i}(d_{hi}-\bar{d})^2}$$

$$\textit{Sstress} = \sqrt{\sum_{h,i}(d_{hi}-\hat{d}_{hi}^2)^2}$$

The denominators in the two stress formulas are scaling terms that make the objective functions dimensionless and produce stress values between 0 and 1. All are relative, measuring the decrease in lack of fit between iterations.

NMDS: Step-6

Improve the configuration by moving it slightly in a direction of decreasing stress. This is done by a numerical optimization algorithm called *method of steepest descent*.

This direction is found by analyzing the partial derivatives of the stress function (Carroll 1987). The idea is to move the points in the ordination plot to new positions that are likely to decrease the stress most rapidly.

NMDS: Steps-7&8

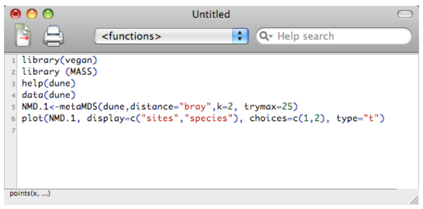
Step-7: Repeat steps 3 to 6 until the objective function reaches a small predetermined value, or until convergence is achieved.

The coordinates calculated at the last passage through step 6 become the coordinates of the *n* objects in the *m* dimensions of the multidimensional scaling ordination.

Step-8: Most NMDS programs then rotate the final solution using PCO, for easier interpretation.

NMDS Example Using R

Let us look at an example from Dutch dune meadows where there are cover values of 30 species across 20 sites.

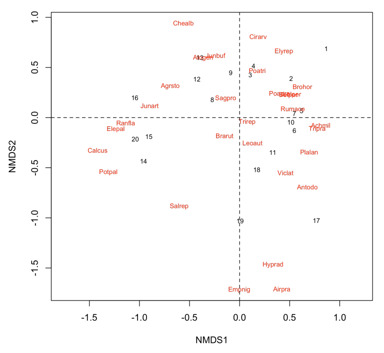


```
library(vegan)
library (MASS)
help(dune)
data(dune)
NMD.1<-metaMDS(dune,distance="bray",k=2, trymax=25)
plot(NMD.1, display=c("sites","species"), choices=c(1,2), type="t")
```

points(x, ...)

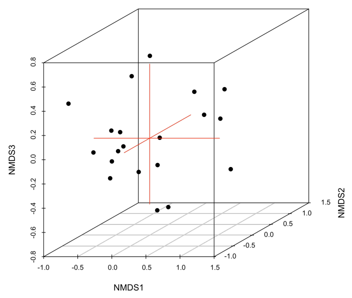
NMDS Example Using R

```
> NMD.1

Call:
metaMDS(comm = dune, distance = "bray", k = 2, trymax = 25)

Nonmetric Multidimensional Scaling using isoMDS (MASS package)

Data:     dune
Distance: bray

Dimensions: 2
Stress:     11.97273
Two convergent solutions found after 6 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'dune'
```

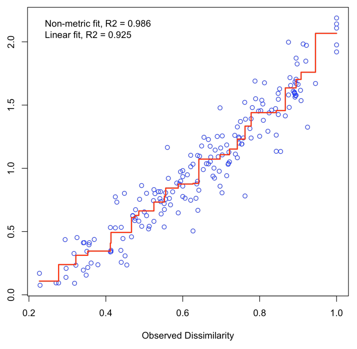


NMDS

```
library(scatterplot3d)
NMD.2<-metaMDS(dune,distance="bray",k=3, trymax=25)
ordiplot3d(NMD.2,choices=1:3, pch=19)
```



NMDS



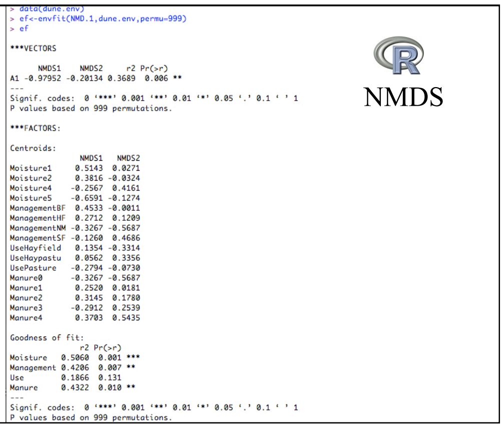
NMDS

> stressplot(NMD.1)

Produces a Shepard plot {vegan}.

Goodness of fit is excellent (> 0.9).

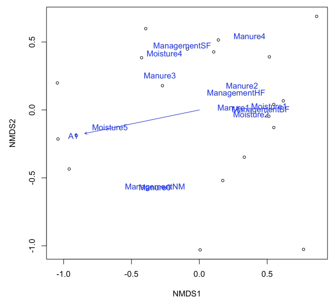
NMDS



```
> data(dune.env)
> ef<-envfit(NMD.1,dune.env,permu=999)
> ef

***VECTORS

      NMDS1    NMDS2     r2 Pr(>r)
A1 -0.97952 -0.20134 0.3689  0.006 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
P values based on 999 permutations.

***FACTORS:

Centroids:
               NMDS1   NMDS2
Moisture1     0.5143  0.0271
Moisture2     0.3816 -0.0324
Moisture4    -0.2567  0.4161
Moisture5    -0.6591 -0.1274
ManagementBF  0.4533 -0.0011
ManagementHF  0.2712  0.1209
ManagementNM -0.3267 -0.5687
ManagementSF -0.1260  0.4686
UseHayfield   0.1354 -0.3314
UseHaypastu   0.0562  0.3356
UsePasture   -0.2794 -0.0730
Manure0      -0.3267 -0.5687
Manure1       0.2520  0.0181
Manure2       0.3145  0.1780
Manure3      -0.2912  0.2539
Manure4       0.3703  0.5435

Goodness of fit:
               r2 Pr(>r)
Moisture   0.5060  0.001 ***
Management 0.4206  0.007 **
Use        0.1866  0.131
Manure     0.4322  0.010 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
P values based on 999 permutations.
```

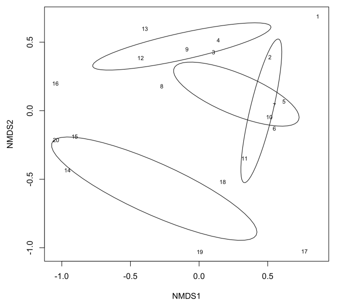
NMDS

```
data(dune.env)
ef<-envfit(NMD.1,dune.env,permu=999)
ef
plot(NMD.1, display="sites")
plot(ef, p.max=0.1)
```

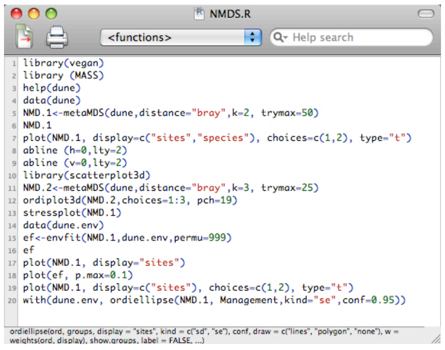


13

```
plot(NMD.1, display="sites")
plot(ef, p.max=0.1)
plot(NMD.1, display=c("sites"), choices=c(1,2), type="t")
with(dune.env, ordiellipse(NMD.1, Management,kind="se",conf=0.95))
```

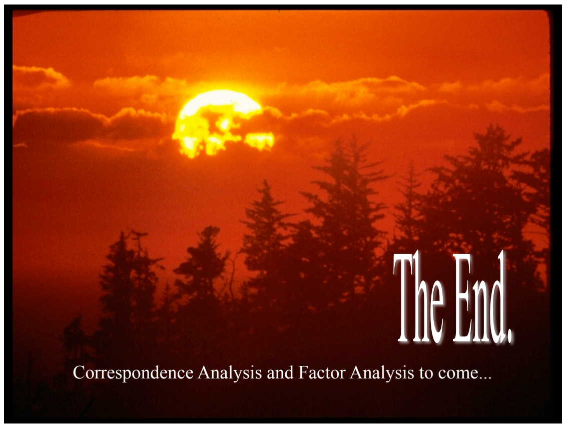


NMDS



```
library(vegan)
library (MASS)
help(dune)
data(dune)
NMD.1<-metaMDS(dune,distance="bray",k=2, trymax=50)
NMD.1
plot(NMD.1, display=c("sites","species"), choices=c(1,2), type="t")
abline (h=0,lty=2)
abline (v=0,lty=2)
library(scatterplot3d)
NMD.2<-metaMDS(dune,distance="bray",k=3, trymax=25)
ordiplot3d(NMD.2,choices=1:3, pch=19)
stressplot(NMD.1)
data(dune.env)
ef<-envfit(NMD.1,dune.env,permu=999)
ef
plot(NMD.1, display="sites")
plot(ef, p.max=0.1)
plot(NMD.1, display=c("sites"), choices=c(1,2), type="t")
with(dune.env, ordiellipse(NMD.1, Management,kind="se",conf=0.95))
```

```
ordiellipse(ord, groups, display = "sites", kind = c("sd", "se"), conf, draw = c("lines", "polygon", "none"), w =
weights(ord, display), show.groups, label = FALSE, ...)
```



The End.

Correspondence Analysis and Factor Analysis to come...