Multivariate Ordination

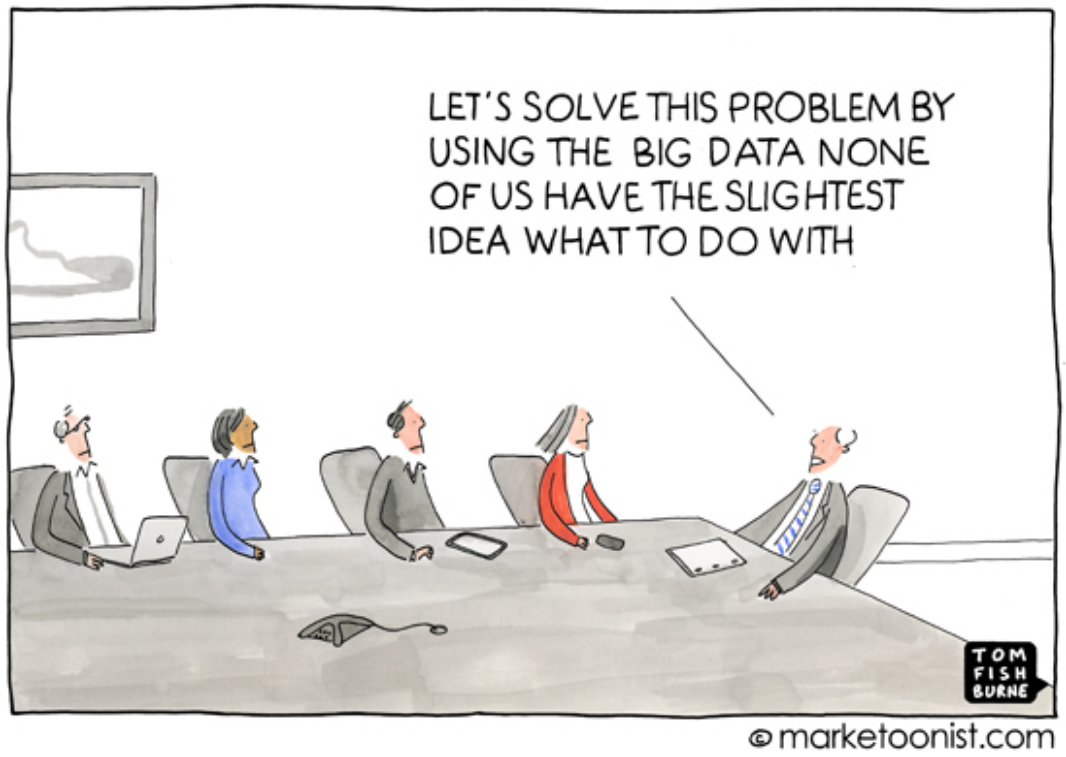
Open Source for Open Science 2017

Michelle Chrpa

Where are we going?

- ❑What is Ordination
- ❑Types of ordination
- ❑Vegan
- ❑Basic Ordination
- ❑Distance Metrics & Dissimilarities
- ❑Principal Coordinates Analysis
- ❑Non-Metric Multidimensional Scaling
- ❑Resources
- ❑Exercise

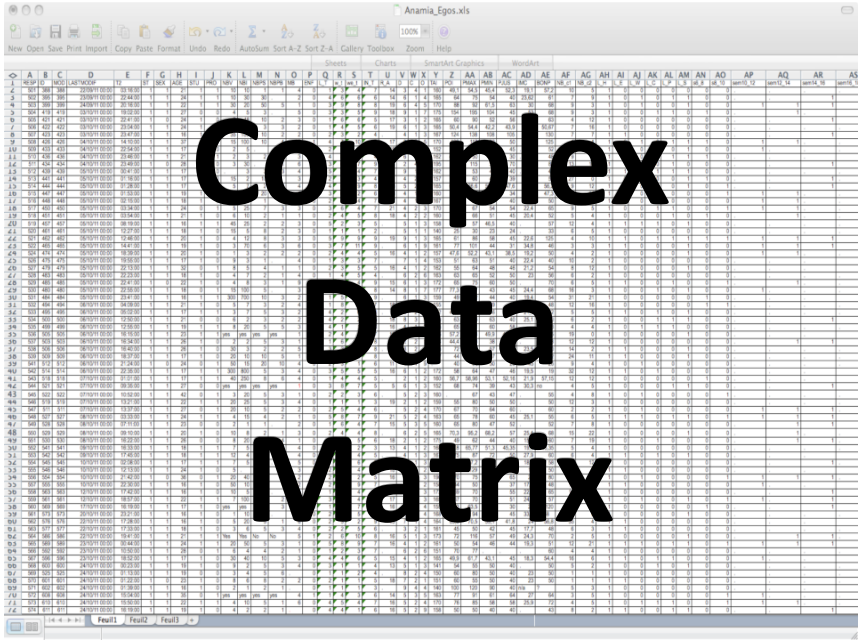
Multivariate Ordination
What is it good for?



What is multivariate?

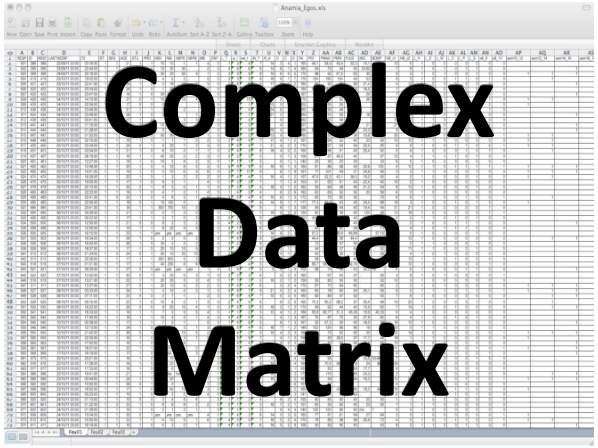
- Multiple variables:
  - Environmental conditions
  - Measured traits
  - Abundance of species

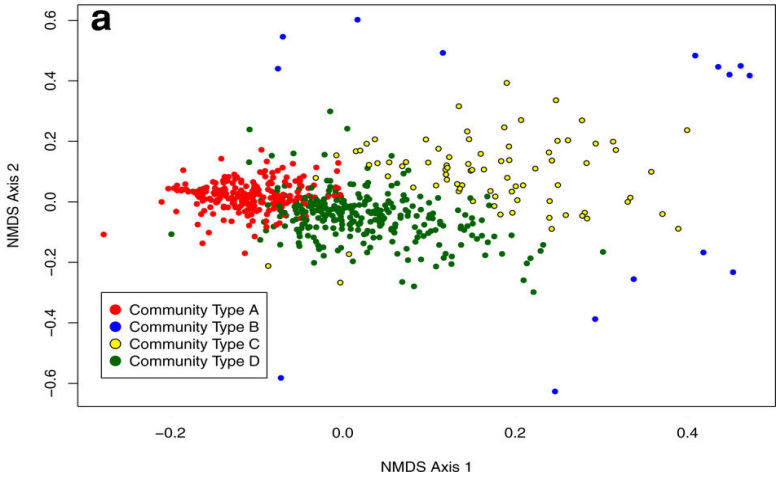
Each variable is an axis



What is ordination?

- Putting things in order using distance
- Distance is relationship, similarity, differeneces
- Ordination uses graphs to visualize complex data in few dimensions
- Find the patterns and combinations of variables that can be used in subsequent analysis
- The goal is to find the greatest variability in community composition for a set of samples



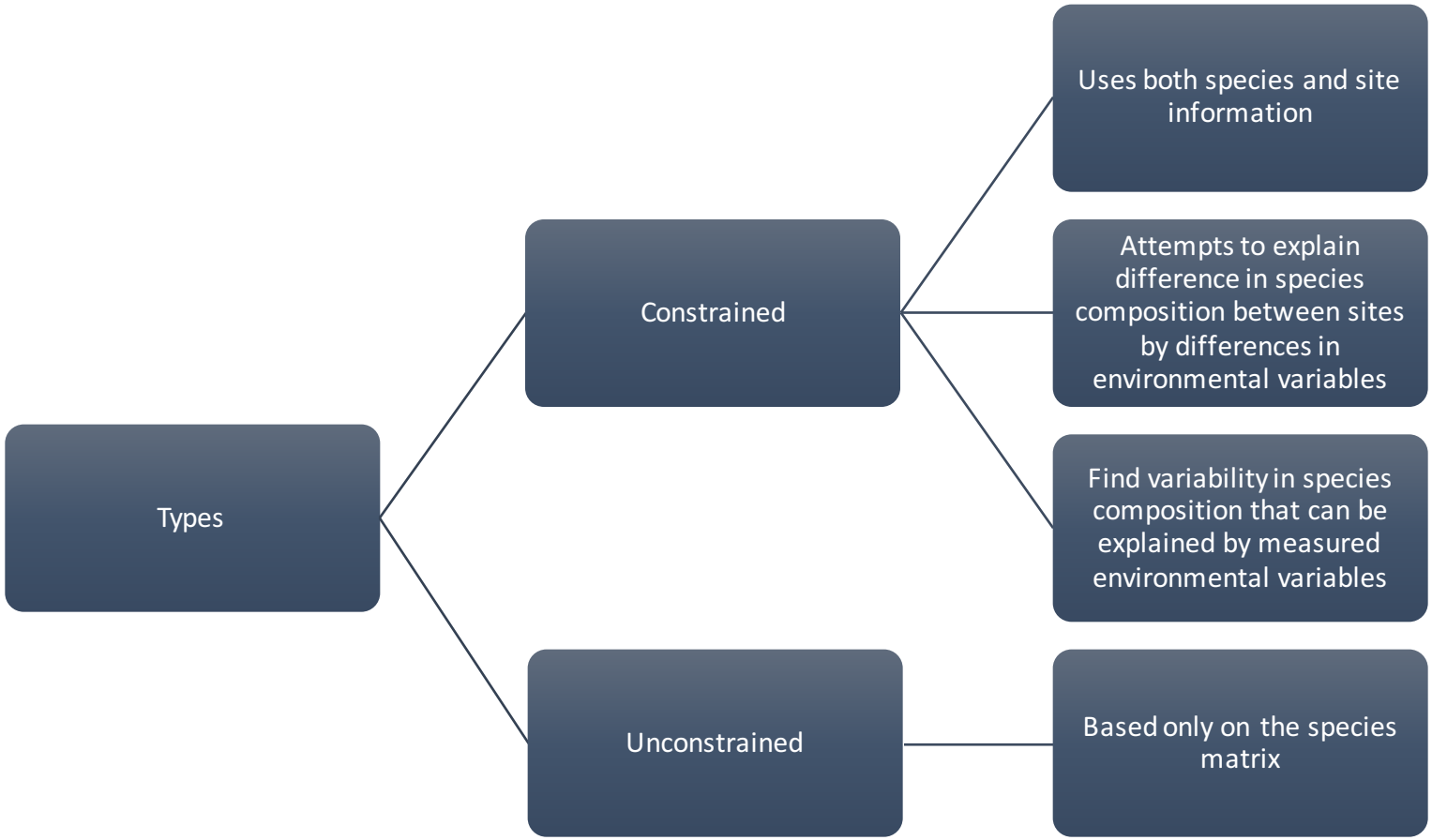


What is ordination?

- A multivariate analysis method
- Data reduction technique
- Applied to data with lots of correlations, helps you make sense of these correlations

- **Structure of data that goes into an ordination!**
  - Any data set that has a series of different variables collected on the same sample units
  - Examples:
    - Species cover in plots (species matrix)
    - Environmental data collected in plots (environmental matrix)
    - Demographic data collected on cities
    - Any set of data collected from a set of spatial points or set of "things" (such as water chemistry data from a set of lakes)

Ordination Types



Multivariate Ordination Analysis

Unconstrained

- Principal Component Analysis (PCA)
- Correspondence Analysis (CA)
- Non-Metric Multidimensional Scaling (NMDS)
- Principal Coordinates Analysis (PCOA, MDS)
- Discriminate Analysis (DA)

Constrained

- Redundancy Discriminate Analysis (RDA)
- Canonical Correspondence Analysis (CCA)
- Distance-based redundancy analysis (db-RDA)
- Canonical Analysis of Principal Coordinates (CAP)

Vegan

- The majority of vegan functions work with a single vector, or more commonly an entire data frame
- This data frame may contain the species abundances
- Where subsidiary data is used/required, these two are supplied as data frames
- For example; the environmental constraints in a CCA
- It is not a problem if you have all your data in a single file/object; just subset it into two data frames after reading it into R

Vegan

- As with any package, vegan needs to be loaded into the R session before it can be used
- When vegan loads it displays a simple message showing the version number

Note also that vegan depends upon the permute package

>install.packages("vegan")

>library(vegan)

Simple vegan usage; basic ordination

- First we start with a simple correspondence analysis (CA) to illustrate the basic features
- We are using of the in-built data sets on lichen pastures
- For various reasons to fit a CA we use the cca() function
- Store the fitted CA in ca1 and print it to view the results

```
> data(varespec)
> ca1 <- cca(varespec)
> ca1
```

```
Call: cca(X = varespec)

              Inertia Rank
Total           2.083
Unconstrained   2.083   23
Inertia is mean squared contingency coefficient

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
0.5249 0.3568 0.2344 0.1955 0.1776 0.1216 0.1155 0.0889
(Showed only 8 of all 23 unconstrained eigenvalues)
```

Simple vegan usage; basic ordination

> *class(ca1)*

```
[1] "cca"
```

> *str(ca1, max = 1)*

```
List of 10
 $ call       : language cca(X = varespec)
 $ grand.total: num 2418
 $ rowsum     : Named num [1:24] 0.0369 0.0372 0.039 0.052 0.0374 ...
  ..- attr(*, "names")= chr [1:24] "18" "15" "24" "27" ...
 $ colsum     : Named num [1:44] 0.01864 0.06287 0.00347 0.02097 0.11376 ...
  ..- attr(*, "names")= chr [1:44] "Cal.vul" "Emp.nig" "Led.pal" "Vac.myr" ...
 $ tot.chi    : num 2.08
 $ pCCA       : NULL
 $ CCA        : NULL
 $ CA         :List of 8
 $ method     : chr "cca"
 $ inertia    : chr "mean squared contingency coefficient"
 - attr(*, "class")= chr "cca"
```

The CCA object ?cca.object

- Objects of class "cca" are complex with many components
- Entire class described in ?cca.object
- Depending on what analysis performed some components may be NULL
- Used for (C)CA, PCA, RDA, and CAP (capscale())
- ca1 has:
  - $call how the function was called
  - $grand.total in (C)CA sum of rowsum
  - $rowsum the row sums
  - $colsum the column sums
  - $tot.chi total inertia, sum of Eigenvalues
  - $pCCA Conditioned (partialled out) components
  - $CCA Constrained components
  - $CA Unconstrained components
  - $method Ordination method used
  - $inertia Description of what inertia is

The CCA object ?cca.object

- The $pCCA $CCA, and $CA components contain a number of other components
- Most usefully the Eigenvalues are found here plus the species (variables) and site (samples) score
- ca1$CA has:
  - eig the Eigenvalues (λ)
  - $u (weighted) orthonormal site scores
  - $v (weighted) orthonormal species scores
  - $u.eig u scaled by λ
  - $v.eig v scaled by λ
  - $rank the rank or dimension of component (number of axes)
  - $tot.chi sum of λ for this component
  - $Xbar the standardised data matrix after previous stages of analysis
- *.eig may disappear in a future version of vegan
- There are many other components that may be present in more complex analyses (e.g. CCA)

Extractor functions — eigenvals()

- Thankfully we don't need to remember all those components in general use — extractor functions
- To extract the eigenvalues use eigenvals():

> eigenvals(ca1)

```
      CA1       CA2       CA3       CA4       CA5       CA6       CA7       CA8
0.5249320 0.3567980 0.2344375 0.1954632 0.1776197 0.1215603 0.1154922 0.0889385
      CA9      CA10      CA11      CA12      CA13      CA14      CA15      CA16
0.0731751 0.0575174 0.0443421 0.0254635 0.0171030 0.0148963 0.0101598 0.0078298
     CA17      CA18      CA19      CA20      CA21      CA22      CA23
0.0060323 0.0040079 0.0028650 0.0019275 0.0018074 0.0005864 0.0002434
```

Extractor functions — scores()

- The scores() function is an important extractor if you want to access any of the results for use elsewhere
- Takes an ordination object as the first argument
- choices: which axes to return scores for, defaults to c(1,2)
- display: character vector of the type(s) of scores to return

> str(scores(ca1, choices = 1:4, display = c("species","sites")))

```
List of 2
 $ species: num [1:44, 1:4] 0.022 0.0544 0.8008 1.0589 0.1064 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:44] "Cal.vul" "Emp.nig" "Led.pal" "Vac.myr" ...
  .. ..$ : chr [1:4] "CA1" "CA2" "CA3" "CA4"
 $ sites  : num [1:24, 1:4] -0.149 0.962 1.363 1.176 0.497 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:24] "18" "15" "24" "27" ...
  .. ..$ : chr [1:4] "CA1" "CA2" "CA3" "CA4"
```

> head(scores(ca1, choices = 1:2, display = "sites"))

```
                CA1         CA2
18 -0.149231732 -0.89909538
15  0.962176641 -0.24176673
24  1.363110128  0.25182197
27  1.175623286  0.83540787
```

scores() & scaling in cca(), rda()

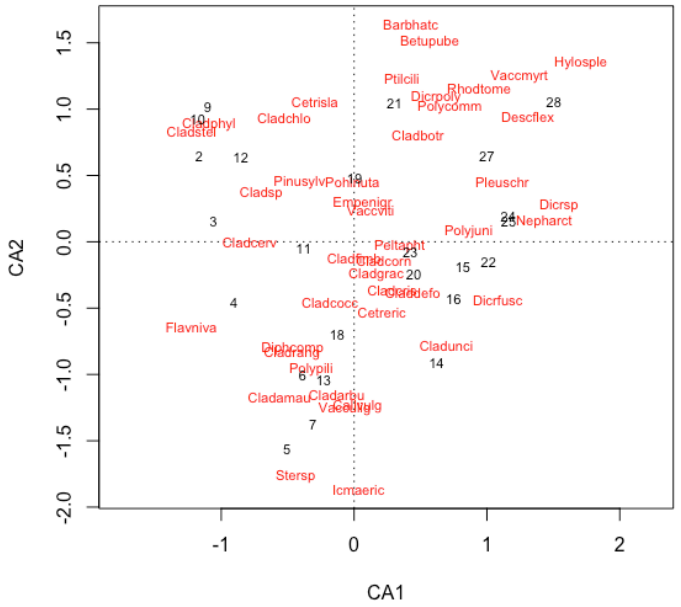
- When we draw the results of many ordinations we display 2 or more sets of data
- Can't display all of these and maintain relationships between the scores
- Solution; scale one set of scores relative to the other
- Controlled via the scaling argument
  - scaling = 1 — Focus on species, scale site scores by λi
  - scaling = 2 — Focus on sites, scale species scores by λi
  - scaling = 3 — Symmetric scaling, scale both scores by √λi
  - scaling = -1 — As above, but □
  - scaling = -2 — For cca() multiply results by (1/(1 − λi))
  - scaling = -3 — this is Hill's scaling
  - scaling < 0 — For rda() divide species scores by species' σ
  - scaling = 0 — raw scores

> scores(ca1, choices = 1:2, display = "species", scaling = 3)

Basic ordination plots

- Basic plotting can be done using the plot() method
- choices = 1:2 — select which axes to plot
- scaling = 3 — scaling to use
- display = c("sites","species") — which scores (default is both)
- type = "text" — display scores using labels or points ("points")
- Other graphics arguments can be supplied but the apply for all scores
- See ?plot.cca for details

> plot(ca1, scaling = 3)



Distance Metrics

- **Euclidean** – straight line distance
- **Bray-Curtis** – dissimilarities between sites
- **Jaccard** – dissimilarity between sample sets
- **Gower** – distances between pairs of variables over two data sets and then combines those distances to a single value per record-pair
- **Kulczynski** – average conditional probability that a characteristic is present in one item given that the characteristic is present in the other item
- **Hellinger** – approximate measurement of the amount of overlap between two statistical samples
- **Chi-Square** – assessing the goodness of fit between observed values and those expected theoretically

Dissimilarities

- dist() is the basic R function for computing dissimilarity or distance matrices Few, if any, of the included metrics are suitable for community ecology data
- Vegan provides vegdist() as a drop-in alternative with numerous useful metrics
  - Bray-Curtis
  - Jaccard
  - Gower
  - Kulczynski
  - Give good gradient separation for ecological data
- Returns an object of class "dist" which can be used in many other R functions & packages

```
> dis <- vegdist(varespec, method = "bray")
> dis2 <- vegdist(varespec, method = "gower")
> class(dis)
[1] "dist"
```

Ecologically meaningful transformations

- Legendre & Gallagher (Oecologia, 2001) show that many ecologically useful dissimilarities are in Euclidean form
- They are equivalent to calculating the Euclidean distance on transformed data
- Two of the suggested metrics included in vegan's decostand() function
  - Chi-square
  - Hellinger

```
> dis3 <- vegdist(decostand(varespec, method = "hellinger"))
```

Principal coordinates analysis

- Principal Coordinates Analysis (PCoA) (AKA classic or metric multidimensional scaling) is PCA applied to a dissimilarity matrix Several functions in R available for this (e.g. cmdscale()) Vegan has capscale() for constrained analysis of principal coordinates (CAP)
- Works exactly the same way as rda() Can supply a dissimilarity matrix as the response or the community
- data and tell vegan which dissimilarity to compute Several new arguments

  distance & dfun indicate which dissimilarity to compute and which function to use to compute it

  sqrt.dist & add allow handling of negative λ (not needed but can be used)

  comm the community data is a dissimilarity matrix used in the model formula; allows species scores to be added

  metaMDSdist process the community data before analysis using metaMDSdist(); allows transformations & extended dissimilarities

Principal coordinates analysis

To use capscale() for PCoA provide a formula with an intercept only LHS of formula is community data or dissimilarity matrix

>(pcoa <- capscale(varespec ~ 1, dist = "bray", metaMDS = TRUE))

```
Square root transformation
Wisconsin double standardization
Call: capscale(formula = varespec ~ 1, distance = "bray", metaMDSdist =
TRUE)

               Inertia Rank
Total          2.54753
Real Total     2.59500
Unconstrained  2.59500   19
Imaginary     -0.04747    4
Inertia is squared Bray distance

Eigenvalues for unconstrained axes:
  MDS1   MDS2   MDS3   MDS4   MDS5   MDS6   MDS7   MDS8
0.6075 0.3820 0.3335 0.2046 0.1731 0.1684 0.1505 0.1163
(Showed only 8 of all 19 unconstrained eigenvalues)

metaMDSdist transformed data: wisconsin(sqrt(varespec))
```

> class(pcoa)

```
[1] "capscale" "rda"      "cca"
```

Non-Metric Multidimensional Scaling

- Aim is to find a low-dimensional mapping of dissimilarities
- Similar idea to PCoA, but does not use the actual dissimilarities
- NMDS attempts to find a low-dimensional mapping that preserves as best as possible the rank order of the original dissimilarities (d_{ij})
- Solution with minimal stress is sought; a measure of how well the NMDS mapping fits the d_{ij}
- Stress is sum of squared residuals of monotonic regression between distances in NMDS space (d^*_{ij}) & d_{ij}
- Non-linear regression can cope with non-linear responses in species data
- Iterative solution; convergence is not guaranteed Must solve separately different dimensionality solutions

Non-Metric Multidimensional Scaling

- Use an appropriate dissimilarity metric that gives good gradient separation rankindex()
  - Bray-Curtis
  - Jaccard
  - Kulczynski
- Wisconsin transformation useful; Standardize species to equal maxima, then sites to equal totals wisconsin() Iterative solution; use many random starts and look at the fits with lowest stress
- Only conclude solution reached if lowest stress solutions are similar (Procrsutes rotation) Fit NMDS for 1, 2, 3, . . . dimensions; stop after a sudden drop in stress observed in a screeplot
- NMDS solutions can be rotated at will; common to rotate to principal components Also scale axes in half-change units; samples separated by a distance of 1 correspond, on average, to a 50% turnover in composition

NMDS in vegan

Vegan implements all these ideas via the metaMDS() wrapper

> library("vegan"); data(dune)

> set.seed(42)

> (sol <- metaMDS(dune))

```
Run 0 stress 0.1192678
Run 1 stress 0.1183186
... New best solution
... procrustes: rmse 0.02026936  max resid 0.06495232
Run 2 stress 0.1192678
Run 3 stress 0.1183186
... procrustes: rmse 5.509367e-06  max resid 1.511849e-05
*** Solution reached

Call:
metaMDS(comm = dune)

global Multidimensional Scaling using monoMDS

Data:     dune
Distance: bray

Dimensions: 2
Stress:     0.1183186
Stress type 1, weak ties
Two convergent solutions found after 3 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'dune'
```

NMDS in vegan

If no convergent solutions, continue iterations from previous best solution

> (sol <- metaMDS(dune, previous.best = sol))

```
Starting from 2-dimensional configuration
Run 0 stress 0.1183186
Run 1 stress 0.1183186
... procrustes: rmse 1.919781e-05  max resid 6.3831e-05
*** Solution reached

Call:
metaMDS(comm = dune, previous.best = sol)

global Multidimensional Scaling using monoMDS

Data:     dune
Distance: bray

Dimensions: 2
Stress:     0.1183186
Stress type 1, weak ties
Two convergent solutions found after 4 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'dune'
```

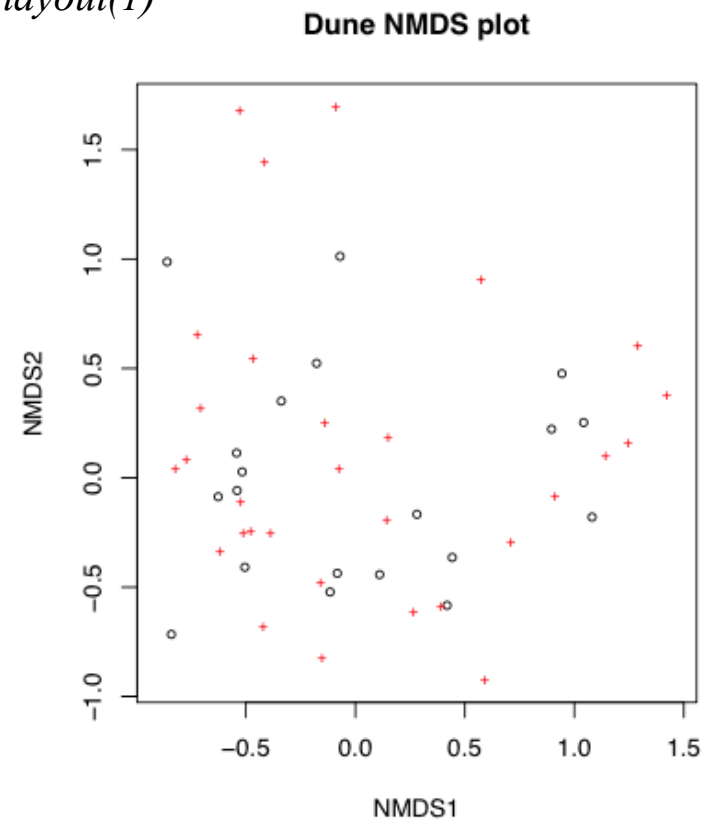
NMDS in vegan

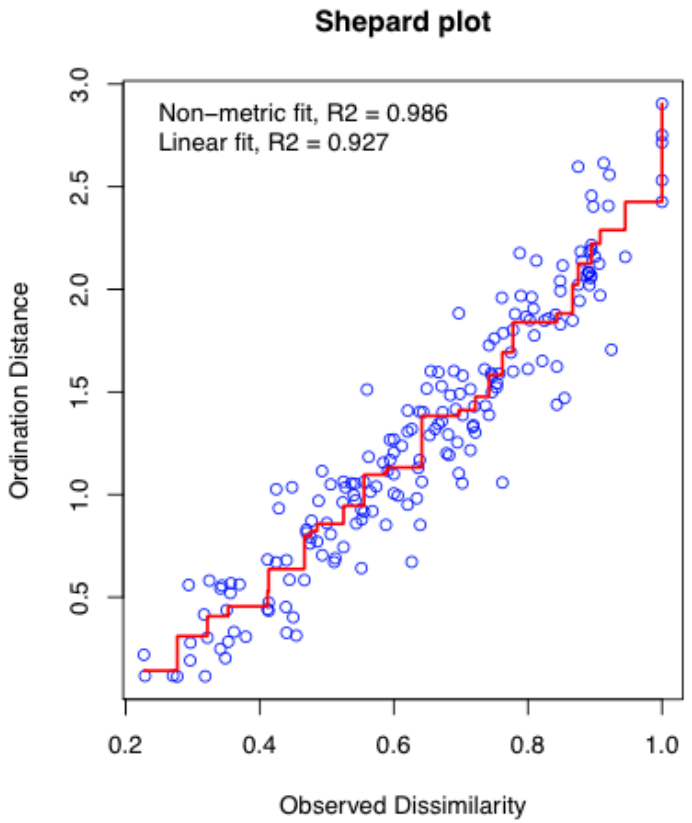
> *layout(matrix(1:2, ncol = 2))*

> *plot(sol, main = "Dune NMDS plot")*

> *stressplot(sol, main = "Shepard plot")*

> *layout(1)*





(A few) Useful Resources

- Vegan tutorial:
  http://cc.oulu.fi/~jarioksa/opetus/metodi/vegantutor.pdf
- The little book of r for multivariate analyses:
  http://little-book-of-r-for-multivariate-analysis.readthedocs.org/en/latest/src/multivariateanalysis.html#means-and-variances-per-group
- Community analyses lectures by Jari Oksanen:
  http://cc.oulu.fi/~jarioksa/opetus/metodi/
- Ordination Methods by Michael Palmer:
  http://ordination.okstate.edu/overview.htm#Nonmetric_Multidimensional_Scaling

Exercises

- **Vegan: An Introduction To Ordination**, Jari Oksanen

https://cran.r-project.org/web/packages/vegan/vignettes/intro-vegan.pdf

- **Little Book of R for Multivariate Analysis,** Avril Coghlan

https://little-book-of-r-for-multivariate-analysis.readthedocs.io/en/latest/