Data Analysis in Paleontology Using *R*

Session 3

24 Jan 2006

Gene Hunt
Dept. of Paleobiology
NMNH, SI

Data Manipulation

Sorting

```
x <- c(3,1,10)
sort(x)   # sorts vector (does not replace x)
order(x)  # gives sorted indices
rank(x)   # gives ranks (averages ties)
```

Selecting

```
subset(data)  # subsets cases/variables of a dataframe
which(tf)     # gives indices for which tf is TRUE
which(valve.length<600)   # indices of small populations
```

Combining

```
rbind()  # combines rows of vector/matrix
cbind()  # combines columns of vector/matrix
merge(d1, d2)  # dataframe join (like databases)
```

Data Manipulation

Tabulating

```
table(f1)        # counts per unique value of f1
table(f1, f2)    # cross tabulation
```

```
data(mtcars)       # car data again
attach(mtcars)
table (cyl)
table (cyl, am)  # tabulate number cylinders vs.
                 # transmission type (am=1 is auto)
```

```
xtabs(formula)     # does same with formula interface
     # also useful if data are already cross-tabulated
     # (e.g., database query output)
     # has summary() and plot() methods
```

Manipulating Matrices

Often want to do an operation on every row or column of a matrix

```
apply(X, MARGIN, FUN,…)
   # MARGIN=1 by row; MARGIN=2 is by column
   # FUN is function to be applied
   # … are arguments for FUN (if needed)
   # returns a vector or matrix of results
```

```
apply(X, MARGIN=1,FUN=sum)  # row totals
apply(X, 2, sum)         # column totals
apply(X, 2, mean)        # column (variable) means
```

Manipulating Matrices

Can apply functions split by a factor

```
tapply(x, INDEX, FUN,…)
    # x is a vector
    # INDEX is factor(s) to split by
    # FUN is function to be applied
    # … arguments for FUN (if needed)
```

```
attach(cope)       # cope dataset
tapply(valve.length, INDEX=species, FUN=mean)
      # mean valve.length separately by species
```

See also: `sapply()`, `lapply()`

Looping

- Sometimes need to explicitly loop through series of operations
- Usually use function `for()`

```
for (i in 1:10)  print (i)
for (i in c(5,4,237))  print (i)   # try this

for (i in 1:10)
 {
     # multiple expressions need to be grouped
     # within braces {}
 }
```

```
nr <- nrow(X)                # number of rows in X
row.tot <- array(dim=nr)     # create vector
for (i in 1:nr)
  row.tot[i]<- sum(X[i,])  # same as apply(X,1,sum)
```

Exercise 10. Data Manipulation & Looping

1. Use a loop to gather the data necessary to draw a rarefaction curve of the first sample (row) of the BCI dataset in the vegan package. (hints: 1. find out how many individuals are in this sample, and then generate a sequence of sample sizes you would like to sample (`ns`) using `seq()`. 2. Create an array to hold the rarefied diversities (`rrich`) as they are computed. 3. Loop through `1:length(ns)`, calling `rarefy()` and saving the result to the appropriate element of `rrich`.)
2. Plot the resulting rarefaction curve, giving informative labels to the x and y axes.

What are the relationships among samples/variables?

- Questions like:
  - Which are most similar?
  - Are there underlying groups or gradients?
  - Do patterns relate to extrinsic variables?
- Sometimes divided into two types:
  - R-mode: among variables (colums)
  - Q-mode: among samples (rows)

What are the relationships among samples/variables?

Problem: complicated--too many variables, many of which are correlated

Solution: distill signal into fewer axes that (hopefully) preserve major features of data structure

Usually rely on three steps…

1. Transform data (if desired)
2. Calculate distance matrix
3. Reduce dimensionality using cluster or ordination of distances

Data Transformation

Data matrix = X

1. Transform whole matrix

e.g. `log(X)`, `sqrt(X)`, `asin(X)`

2. Transform matrix rows

- can loop through rows using `for()`, or use `apply()`
- special case: absolute to relative abundances

```
prop.table(X, margin=1) # X must be a matrix
                        # if not, use as.matrix(X)
```

Data Transformation

Data matrix = X

3. Transform matrix columns

- can loop through columns using `for()`, or use `apply()`
- special case: mean-centering and standardization

```
scale(X, center=TRUE)  # mean-centers
scale(X, center=TRUE, scale=TRUE) # also sd=1
```

`decostand(X, method=)` does most ecological transformations

```
method="total"     # converts to relative abundance
method="pa"        # converts to presence-absence
```

Computing Distances

Functions

```
dist(X, method="euclidean")
vegdist(X, method="bray")    # in vegan package
```

- Lots of different metrics available (`method` argument)
- Euclidean often used in morphometrics
- Bray-Curtis common (among lots of others) in ecology

$$d_{Bray} = \frac{\sum_i |x_i - y_i|}{\sum_i (x_i + y_i)} \qquad d_{Euc} = \sqrt{\sum_i (x_i - y_i)^2}$$

x and y are vectors (rows of data matrix)

S. Wing's Data

- Floral data from one site (BCR, ~71 Ma)
- In situ preservation, 100 sites along 4km outcrop
- Counts for 130 taxa; sed/environmental var's

```
sw <- read.table(file="BCRDATA.txt", header=T)
```

Columns	Type of data
1:130	Taxon abundances
131:137	Debris abundances
138:147	Sed/env variables

```
X <- sw[,1:130]        # floral data only
carb <- sw$EPERC.CARB    # %carbon
```

S. Wing's Data

Check out data

```
site.tot <- apply(X,1,sum)    # site total abundances
tax.tot <- apply(X,2,sum)    # taxon total abundances
range(site.tot)    100    753.5      # not too bad
hist(log10(tax.tot), col="tan")
```

Transform data, compute distance matrix

```
Xt <- sqrt(X)    # sqrt transform, reduces effect
                 # of dominant taxa
Dbr<- vegdist(Xt, method="bray")
```

Cluster Analysis

Hierarchical, agglomerative methods are generally used

```
w.cl<- hclust(Dbr, method="average") #other methods available
w.cl<- hclust(Dbr, meth="aver") #same as previous
plot(w.cl)
```

Properties of Cluster Analyses

1. Patterns reduced to 1-dimension
2. Short distances preserved, longer distances distorted
3. Yields hierarchical groupings whether present or not

Cluster Analysis

Cophenetic distances (implied by clustering)

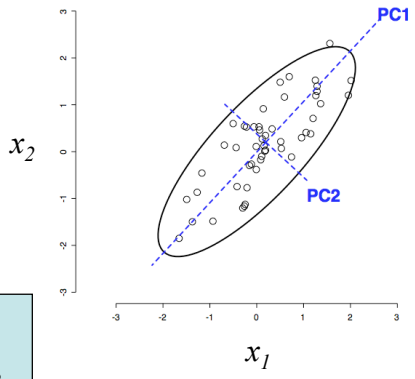
```
dist.cl <- cophenetic(w.cl) # returns distance matrix
plot(Dbr, dist.cl, xlab="True Distance", ylab="Distance
      Implied by Clustering")
abline(0,1, col="red")   # add line y=x
```

There are other hierarchical and non-hierarchical clustering methods available, see:

```
kmeans(), {cluster package}, {mclust package}
```

Principal Components Analysis (PCA)

- Common for biometric data, not in ecology
- New axes based on directions of maximal variance (cov or cor)
- PC axes direction defined by variable loadings (eigenvectors), length defined by eigenvalues (variance)
- PC scores computed for observations



Eigenvectors

	PC1	PC2
x_1	0.68	0.73
x_2	0.73	-0.68

PCA in *R*

Function `prcomp(x, scale.=FALSE)` `# scale.=F uses COV`
`# scale.=T uses COR`

Returns:

```
sdev        # square root of eigenvalues
rotation    # eigenvector coefficients in col's
x           # PC scores
```

```
w.pca<- prcomp(Xt, scale.=TRUE)  # use COR
biplot(w.pca) # biplot (scaled scores & loadings)
screeplot(w.pca, log="y")  # plot of eigenvalues
```

Principal Coordinates Analysis (PCO)

- Like PCA, based on eigenvectors
- Instead of COV or COR matrix, eigen decomposition is performed on a Gower-transformed distance matrix *(Gower, 1966)*
- Useful for mixtures of data types (interval, ordinal, categorical)
- Sometimes called "classical" or "metric" multidimensional scaling
- Common in morphospace studies

In *R*, function `cmdscale()`

```
w.pco<- cmdscale(Dbr, k=2)
     # k is number of PCO axes
```

Correspondence Analysis

- Simultaneous ordination of rows (sites) and columns (taxa) to maximize correlation btwn site and taxon ordinations.
- Preserves χ^2 distance
- Sometimes called Reciprocal Averaging
- Another variant constrains CA based on extrinsic (environmental) variables (CCA)

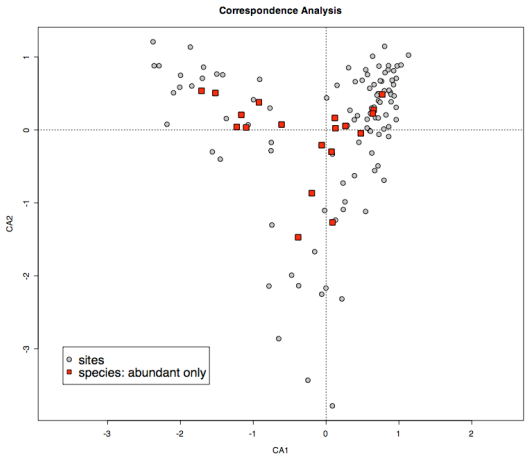
In *R*, function `cca()` from `vegan`

```
w.ca<- cca(Xt)
```

* There is also a function of the same name in package `ade-4`

Ordinations in vegan

- Plot method is a biplot (little control over options)
- Can use `scores(w.ca)` to extract scores

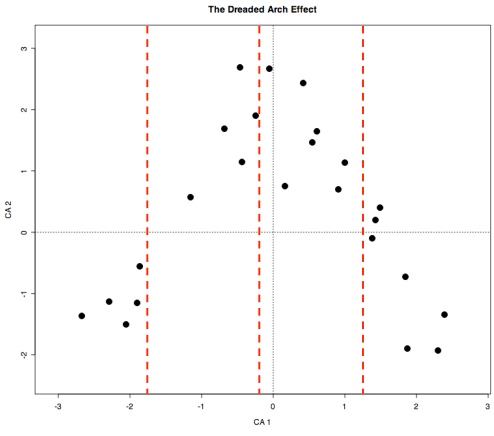


More control

```
fig<- ordiplot(w.ca,
type="n")
points(fig, what="sites",
pch=21, bg="grey", cex=1.5)
points(fig, what="species",
pch=22, cex=2, bg="red",
select=tax.tot>200)
```

Detrended Correspondence Analysis (DCA)

- Modified CA, designed to correct the "arch effect"
- Divided CA1 into segments and centers and scales within each segment
- Thought to be good at detecting a gradient
- In marine paleoecology, DCA1 is usually interpreted as 'depth'




```
w.dca <- decorana(Xt)
```

Nonmetric Multidimensional Scaling (NMDS)

- Seeks reduced dimensionality ordination that maximizes rank-order correlation of inter-point distances (number of dimensions determined ahead of time)
- Iterative algorithm, measure of rank-order misfit is called "stress"
- Considered to be robust--aberrant samples do not dominate axes
- Scaling of axes is arbitrary (as is rotation and translation)

NMDS in *R*

```
w.mds <- isoMDS(Dbr, k=2)    # k is number of axes
                             # returns $points, $stress

symbols(w.mds$points, circles=carb, inches=0.3)
```

See also: `sammon() {MASS}`, `metaMDS() {vegan}`

What do NMDS axes *mean*?

Can compute equivalents of loadings: the correlation coefficient btwn new (MDS) and original variables

```
cor (Xt, w.mds$points)  # mds "loadings"
```

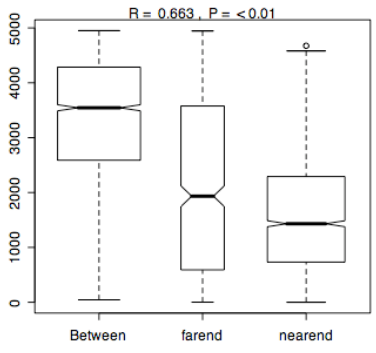
ANOSIM

- Similar to ANOVA in that testing for differences across *a priori* groups (factors).
- Based on a distance matrix: convert pairwise distance to their ranks, and divides them into *within-group* and *between-group* distances.
- Test statistic, *R*, is based on (mean rank between groups) minus the (mean rank within groups).
- *R* can range from -1 to 1, but is almost always positive.
- Significance is assessed using a permutation test

$$R = \frac{\bar{r}_B - \bar{r}_W}{N(N-1)/4}$$

ANOSIM in *R*

```
gg <- array(dim=100)            # grouping variable
gg[sw$EDIST > 3000] <- "farend"
gg[sw$EDIST < 3000] <- "nearend"     # lateral position
aa <- anosim(Dbr, gg, perm=500)
plot(aa)
```



Note: also could test association between `Dbr` and pairwise geographic distance using `mantel() {vegan}`

Exercise 11. Cluster & Ordinations

1. Compare the clustering of `Dbr` based on average linkage to that with `method="single"`. Do the overall shapes of the dendrograms differ? Different linkage algorithms produce characteristically different tree shapes. In particular, note tree balance (symmetry).
2. Compute the eigenvalues of the PCA results, and save them to a variable `ev`. Use this variable to make your own scree plot using `plot()`. Use type="o" or "b" to get overlapping lines and symbols. Plot only the first 10 eigenvalues, and remember to log-scale the y-axis.
3. Using `ev`, compute the proportion of variance explained by the eigenvalues; save this to a variable `pev`. Use the function `cumsum()` to compute the cumulative proportion of variance explained, and call it `cpev` (check ?cumsum). Use `cbind()` to make a matrix of the first ten elements of `ev, pev` and `cpev`.
4. Use the function `scores()` to extract the site scores from the DCA results. Use the function `symbols()` to plot the first two DCA axes, with circle size proportional to `carb`. Are the results consistent with the NMDS ordination?
5. Do the following: `shep <- Shepard(Dbr, w.mds$points)`, and then plot `shep` (when there are lots of points, it sometimes helps to make them very small, as with `cex=0.3`). This is called a Shepard Plot, and shows true pairwise distance between points (on the x-axis) against the pairwise distance in the ordination space. In addition to the names components `x` and `y` (which are plotted), `shep` also has a named component `yf`, which is the monotonic function relating true and ordinating distances used in minimizing stress. Add this to the plot, using: `lines(shep$x, shep$yf, col="red", type="S")`.
6. How similar are the site ordinations in CA and DCA? One way to see the effects of the detrending is to compute the correlation coefficients between the site ordinations. Do this, using `scores()` to extract the site ordination information (note that `scores()` yields both site and species scores for CA, so be careful).

Answers to Exercises [2]

Exercise 7. Factors, types, and missing data

```
1. read.table(file="data.txt",header=T,na.strings="-999")  2. Sum(age>20) # 17
3. median(x, na.rm=TRUE)
```

Exercise 8. Statistical models

```
1. #note that the order of arguments for plot is reversed in model notation
2. plot(fitted(w), resid(w))  # note nonrandom pattern  3. # Note that larger symbols are
   mostly above, and small symbols mostly below, the line.
4. w.td<- lm(valve.length ~ mg.temp + depth) # both terms are significant, residuals look ok
```

Exercise 9. Biodiversity and Additional Exercises

```
1. x<- BCI[1,]; px<- x/sum(x)
2. Normally: H<- -sum(px*log(px))  # returns NaN bc some of the px's are zero and log(0)
   is undefined. Instead: nz<- px>0 # T if px>0; H<- -sum(px[nz]*log(px[nz]))
3. rich<- specnumber(BCI); fa<- fisher.alpha(BCI); shan<- diversity(BCI);
   simp<- diversity(BCI, index="simp")
4. fa & rich, and simp & shan, are highly correlated
5. The F-test is highly significant, indicating that w.td significantly improves the fit
```

Answers to Exercises [3]

Exercise 10. Exercise 10. Data Manipulation & Looping

```
1. ns<- seq(10,480,10); rrich<- array(dim=length(ns));
   for (i in 1:length(ns))    rrich[i]<- rarefy(BCI[1,], ns[i])
2. Plot(ns, rrich, type="o", xlab="Sample size", ylab="Rarefied richness")
```

Exercise 11. Cluster and Ordination

```
1. plot(hclust(Dbr, meth="single")); Single linkage trees tend to be very imbalanced;
     average linkage trees tend to be relatively balanced).
2. ev<- w.pca$sd^2; plot(ev[1:10], type="o", log="y")
3. pev<- ev/sum(ev); cpev<- cumsum(pev); cbind(ev[1:10], pev[1:10], cpev[1:10])
4. ds<- scores(w.dca); symbols(ds[,1], ds[,2], circle=carb, inches=.3); #carb is also
     correlatd with the DCA ordination.
5. plot(shep, pch=19, cex=0.3); note that rank order, not linear correlation is minimized
6. ds<- scores(w.dca); cs<- scores(w.ca); cor(ds, cs$sites); note that the first axes are
     almost identical (except for sign, which is arbitrary), but none of the other axes are
     especially correlated.
```