1. Introduction to R

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

1. What is R
2. Installing R
3. Packages, packages and sources for R
4. Importing and exporting data
5. Variables, Data Structure, Graphing

> **A quick note on font conventions**
>
> All R documents provided in class are typed using Trebuchet MS font. However, when R code is presented, referenced, or when R output is given, I have used 10 point **`Bold Courier New Font`**.

WHAT IS R?

R is a language and environment for statistical computing and graphics, similar to the S language originally developed at Bell Labs. It is an open source solution to data analysis that is supported by a large and active worldwide research community. R is exceptional statistical software for ecological analysis as it includes a broad range of multivariate techniques, as well as numerous routines for exploratory data analysis. It handles and analyzes data very effectively and it contains a suite of operators for data manipulation, simulation and calculation. It also has the graphical capabilities for very sophisticated graphs and data displays. Finally, it is an elegant, object-oriented programming language. Technically, the language is called S, and R is the open source implementation available for many systems for free. The R project was started by Robert Gentleman and Ross Ihaka (that's where the name "R" is derived) in the Statistics Department at the University of Auckland in 1995. The software has quickly gained a wide audience. The R project web page is http://www.r-project.org.

R is a powerful programming environment, but can be intimidating if you know little about programming. There aren't any flashy buttons to press, just an unforgiving, blinking prompt that offers little comfort and no direction. But don't worry; we're in this together. Manuals and tutorials are available at http://cran.r-project.org/other-docs.html.

Luckily for you, most of the multivariate techniques we'll be performing in this class can be performed by R programs and require relatively little understanding of the R programming language. Program authors typically bundle sets of statistical programs into Packages that can be downloaded from the web and called from R. In the following sections we cover some of the basics of R. Learning these fundamentals is essential for FISH 560 because it will help you unleash the power of R for multivariate analysis.

WHY USE R?

R has many features to recommend it:

- Most commercial statistical software platforms cost thousands, if not tens of thousands of dollars. R is free! If you are a teacher or a student, the benefits are obvious
- R runs on a wide variety of platforms including Windows, UNIX and Mac OS X.
- R is a comprehensive statistical platform, offering all manner of data analytic techniques.
- R has state-of-the-art graphics capabilities.
- R provides an unparalleled platform for programming statistical methods in an easy manner.
- R contains advanced statistical routines not yet available in other packages.

OBTAINING AND INSTALLING R

R is freely available from the Comprehensive R Archive Network (CRAN) at http://www.r-project.org. Precompiled binaries are available for Linux, Mac OS X, and Windows. Follow directions for installing the base product on the platform of your choice. Later we'll talk about adding additional functionality through optional modules called packages (also available from CRAN).

LIBRARIES AND PACKAGES FOR MULTIVARIATE ANALYSIS

R comes with extensive capabilities right out of the box. However, some of its most exciting features are available as optional modules that you can download and install. There are thousands of user contributed modules called packages that you can download from the CRAN website. They provide a tremendous range of new capabilities, from the analysis of geostatistical data to multivariate data! We will use many of these optional packages in FSH 560.

What are packages?

Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored on your computer is called the library. The function .libPaths() will show you where your library is located, while the function library() will show you what packages you have saved in your library.

Installing a package

There are a number of R functions that let you manipulate packages. To install a package for the first time, use the install.packages() command or click on the *packages* pull down menu at the top of the R program. For example, install.packages() without options will bring up a list of CRAN mirror sites.

Once you select a site, you will be presented with a list of all available packages. Selecting one will download and install it. You only need to install a package once. However, like any software, packages are often updated by their authors. Use the command update.packages() to update any packages that you have installed. To see details on your packages, you can use the installed.packages() command.

Loading a package

Installing a package downloads it from a CRAN mirror site and places it in your library. To actually use it in an R session, you need to load the package using the library() command. For example, to use the packaged vegan, issue the command library(vegan). Of course, you must have installed a package before you can load it. You will have to load a package once in each session you want to use it.

At present, there are a number of libraries or packages available specifically for multivariate analyses typically conducted by ecologists, including VEGAN from Jari Oksanen and LABDSV from Dave Roberts. These libraries are available at CRAN website and can be installed using the instructions above. We will make extensive use of them in subsequent lectures/labs, so be sure to install these packages at the beginning of each lab.

Learning about a package

When you load a package, a new set of functions and datasets become available. Small illustrative datasets are provided along with sample code, allowing you to try out the new functionalities. The help system contains a description of each function (along with examples), and information on each dataset included. Entering help(package="name") will provide a brief description of the package named and an index of the functions and datasets included. Using help()with any of these function or dataset names will provide further details. The same information can be downloaded as a PDF manual from CRAN.

SOURCES FOR MULTIVARIATE ANALYSIS

Biostats.R is a suite of R functions written by Kevin McGarigal (University of Massachusetts) to facilitate the analysis of multivariate data sets. This is a pseudo-library that contains a set of functions that must be sourced by clicking on the *File* menu at the top, and then clicking on *Source R code* …. Locate the

file Biostats.R from your directory. Help documentation has been provided. The Biostats.R file is available on the course's Canvas website.

Common mistakes in R programming

There are some common mistakes made frequently by both beginning and experienced R programmers. If your program generates an error be sure the check for the following:

Using the wrong case. **`help()`**, **`Help()`**, and **`HELP()`** are three different functions (only the first will work).

Forgetting to use quote marks when they are needed. **`install.packages("vegan")`** will work, while **`install.packages(vegan)`** will generate an error.

Forgetting to include the parentheses in a function call. **`help()`** rather than **`help`**. Even if there are no options, you still need the ().

Using the \ in a path name on Windows. R sees the backslash character as an escape character. **`setwd("c:\mydata")`** will generate an error. Use **`setwd("c:/mydata")`** or **`setwd("c:\\mydata")`** instead.

Using a function from a package that is not loaded. The function **`hclust()`** is contained in the vegan package. If you try to use it before loading the package, you will get an error.

The error messages in R can be cryptic, but if you are careful to follow the points above, you should avoid seeing many of them.

IMPORTING DATA INTO R

Getting data into any program is often the hardest part about using the program. For R, this is generally not true, as long as the data are reasonably formatted. The R Development Core Team has developed a special manual to cover the ins and outs of getting data into and out of R. It's available as a PDF or HTML at http://cran.r-project.org.

The first thing to do before importing any file is to tell R what directory your file is in. Do this by going under the File menu and choosing the "Change dir" option. Type in or browse to the directory of your data file and press the OK button.

The easiest way to format the data is in columns, with column headings, and blanks or tabs between. For example:

```
sample elev aspect slope river
   1 1300   240   30   skagit
   2 1640   170   20   cowlitz
   3 1840    NA   24   quinault
   .  .     .     .    .
   .  .     .     .    .
   .  .     .     .    .
 100 1730    70   15    skagit
```

The columns do not need to be straight, but multi-word variables need to be connected or put in quotes. The R convention (but it is just a convention) is to connect with a period. It CANNOT be

connected with "$". Recent versions of R allow connections with "_", but this will cause problems with older versions of R. The above file (if named "site.dat" for instance) could be read with the read.table command as follows:

```
site <- read.table('site.dat',header=TRUE)
```

The resulting data frame would be named "site", and the columns would be named exactly as in the data file. Note that the value for aspect in the third sample is NA. This is a missing value code, and will cause R to treat that value as missing, rather than as a code NA. It's possible to use other codes as missing values if you specify them in the read.table command. For example, suppose in your data set you used -999 as the missing value code. To tell R to set -999 to missing, add the na.strings= argument as follows:

```
site <- read.table('site.dat',header=TRUE,na.strings="-999")
```

Preferably, data can be organized as a "csv" comma delimited file, as follows:

```
sample,elev,aspect,slope,river
1,1300,240,30,skagit
2,1640,170,20,cowlitz
3,1840,90,24,quinault
. . . . .
. . . . .
. . . . .
100,1730,70,15,skagit
```

In which case it would be read:

`site <- read.table('site.dat',header=TRUE,sep=",")` to tell R that the values were separated by commas. Alternatively, you can use:

`site <- read.csv('site.dat',header=TRUE)` to read the file, as read.csv() calls read.table() with the appropriate parameters as defaults.

Finally, if the data are in a formatted file with no delimiters (spaces or commas) it can be read by specifying the columns that start each field. For example:

```
1130024030skagit
2164017020cowlitz
31840 9024quinault
  . .    .   .   .
  . .    .   .   .
  . .    .   .   .
1001730 7015skagit
```

can be read

```
site <- read.table('site.dat',sep=c(1,4,8,11,13))
```

In this case (or in any case where column headings are absent), they can be entered separately with the names command. For example:

```
names(site) <- c("sample","elev","aspect","slope","river")
```

Row names (such as sample IDs) can also be added if desired, using the row.names() function in a similar way.

There is one element of **`read.table()`** that sometimes causes problems. Ordinarily, **`read.table()`** will use the first column that contains all unique values as the row labels. Generally (but not universally) this is the first column. It is often best to explicitly specify which column contains row identifiers (as opposed to data), using **`row.names= specifier`**. Going back to the original example,

`site <- read.csv('site.dat',header=TRUE, row.names=1)` makes sure that R knows that the first column is identifiers, not data.

The beauty of the **`read.table()`** function is the way it handles variables. If any value in a column is alphabetic, it treats the column as composed of "factors," or categorical variables.

Troubleshooting
Sometimes you may have data in a format R will not understand. In such cases using scan () or the data editor to enter your data may be a simple and easy solution. Another trick is to try importing the data into another program, such as a spreadsheet program, and saving it as a different file type. In particular saving spreadsheet data in a text (comma or tab delineated) format is simple and useful. Caution should be used as some spreadsheet programs may restrict the number of data values to be stored.

EXPORTING DATA FROM R

Exporting results from R is usually a less difficult task. The most common method is to write a matrix or data frame to file as a rectangular grid of numbers, possibly with row and column labels. This can be done by the functions write.table and write. Function write just writes out a matrix or vector in a specified number of columns (and transposes a matrix). Function write.table is more convenient, and writes out a data frame (or an object that can be coerced into a data frame) with row and column labels.

USING OUTPUT AS INPUT – REUSING RESULTS

One of the most useful design features of R is that the output of analyses can easily be saved and used as input to additional analyses. Let's walk through an example. If you don't understand the statistics involved, don't worry. We are focusing on the general principle here.

This following code will run a simple linear regression of miles per gallon (mpg) on car weight (wt) using the dataset mtcars. Results are sent to the screen. Nothing is saved.

```
lm(mpg~wt, data=mtcars)
```

This time, the same regression is performed but the results are saved under the name fit.

```
fit <- lm(mpg~wt, data=mtcars)
```

No output is sent to the screen. However, we now can manipulate the results.

The assignment has actually created a list called "fit" that contains a wide range of information from the analysis (including the predicted values, residuals, regression coefficients, and more). Typing **`summary(fit)`** provides details of the analysis, while **`plot(fit)`** produces diagnostic plots. You can generate and save influence statistics with **`cook<-cooks.distance(fit). plot(cook)`** will graph these influence statistics. To predict miles per gallon from car weight in a new set of data use **`predict(fit, mynewdata)`**.

To see what a function returns, look at the Value section of the online help for that function. Here we would look at **`help(lm)`**. This will tell you what is saved when you assign the results of that function to a name.

VARIABLES

Like most programming languages, R allows users to create variables, which are essentially named computer memory. For example, you may store the number of species in a sample in a variable. Variables are identified by a name assigned when they are created. Names should be unique, and long enough to clearly identify the contents of the variable. You may work with the same data weeks or months later, and variable names like x or data are not very helpful. Names can consist of letters, numbers, and the character ".". They may not start with a number, or include the character "$" or any arithmetic symbols as these have special meaning in R.

Variables are assigned a value in an assignment statement, which in R has the variable name to the left of a left-pointing arrow (typed with the "less than" followed by a "dash") with the value behind the arrow. For example,

```
number.species <- 137
```

Recent versions of R allow you to use the = sign for an assignment, i.e. number.species = 137 but I will stick with the older, more elegant arrow.

R allows the creation of variables that contain numeric values (both integers and floating point or real numbers), characters, or special characters interpreted as "logical" values. For example:

```
pi <- 3.14159
small.value <- 1.0e-10
species.name <- 'Homo sapien'
human <- TRUE
```

Notice that real or floating point numbers can be entered with just a decimal point, or in exponential notation, where 1.0e-10 means 1.0^{-10}. Notice also that character variables, called "strings", should be entered in quotes (single or double, it doesn't matter as long as they match). Finally, note that the word TRUE is NOT surrounded by quotes. This is not the WORD TRUE, but rather the VALUE TRUE. Logical variables can only take the values TRUE or FALSE.

Unlike many programming languages (e.g. FORTRAN or C) you do not have to tell R what kind of value (integer, real, or character) a variable will contain; it can tell when the variable is assigned. R will only allow the appropriate operations to be performed on a variable. For example:

```
species.name + 37
Error in species.name + 37: non-numeric argument to binary operator
```

R did not allow 37 to be added to species.name because species.name was a character variable.

DATA STRUCTURES

R is a 4th generation language, meaning that it includes high-level routines for working with data structures, rather than requiring extensive programming by the analyst. There are 4 primary data structures we will use repeatedly.

- **Vectors:** vectors are one-dimensional ordered sets composed of a single data type. Data types include integers, real numbers, and strings (character variables)
- **Matrices**: matrices are two dimensional ordered sets composed of a single data type, equivalent to the concept of matrix in linear algebra.
- **Data frames**: data frames are one to multi-dimensional sets, and can be composed of different data types (although all data in a single column must be of the same type). In addition, each column and row in a data frame may be given a label or name to identify it. Data frames are equivalent to a flat file database, and similar to spreadsheets. Accordingly, we often refer to specific columns in a data frame as "fields."
- **Lists**: lists are compound objects of associated data. Like data frames, they need not contain only a single data type, but can include strings (character variables), numeric variables, and even such things as matrices and data frames. In contrast to data frames, list items do not have a row-column structure, and items need not be the same length; some can be a single value, and others a matrix.

Vectors and Matrices
Vectors, matrices, data frames and lists are identified by a name given the data structure at the time it is created. Names should be unique, and long enough to clearly identify the contents of the structure. Names can consist of letters, numbers, and the character ".". They may not start with a number, or include the character "$" or any arithmetic symbols as these have special meaning in R.

Vectors are often read in as data or produced as the result of analysis, but you can produce one simply using the c() function, which stands for "combine." For example:

`demo.vector <- c(1,4,2,6,12)` produces a vector of length 5 with the values 1, 4, 2, 6, 12.

Individual items within a vector or matrix can be identified by subscript (numbered 1 - n), which is indicated by a number (or numeric variable) within square brackets. For example, if the number of species per sample site is stored in a vector **`spc.site`**, then

`spc.site[37]` = the number of species in site 37

Matrices are specified in the order "row, column", so that

`veg[23,48]` = row 23, column 48 in matrix **`veg`**

Individual rows or columns within a matrix can be referred to by implied subscript, where the value of the desired row or column is specified, but other values are omitted. For example,

`veg[,3]` = third column of matrix **`veg`** represents the third column of matrix **`veg`**, as the row number before the comma was omitted. Similarly,

`veg[5,]` = row 5 of matrix **`veg`** represents row 5, as the column after the comma was omitted. In addition, a number of specialized subscripts can be used.

`veg[]` = all rows and columns of matrix **`veg`**
`spcsite[a:b] = spcsite[a]` through **`spcsite[b]`**
`spcsite[-a]` = all of vector **`spcsite`** except **`spcsite[a]`**
`veg[a:b,c:d]` = a submatrix of **`veg`** from row a to b and column c to d

It's even possible to specify specific subsets of rows and columns that are not adjacent.

`spcsite[c(1,7,10),c(3,6,12)]` = a submatrix consisting of rows 1,7 and 10, and columns 3, 6, and 12 from matrix **`spcsite`**.

Data Frames
Data frames can be accessed exactly as can matrices, but can also be accessed by data frame and column or field name, without knowing the column number for a specific data item. For example, in the MAHA fish dataset (Mid-Atlantic Highlands Area containing environmental characteristics of the fish sampling sites), there is a column labeled "Elev" that holds the elevation of each sample site. This column can be accessed as **`maha$Elev`**, where "maha" is the name of the data frame, "Elev" is the name of the field or column of interest, and the "$" is a separator to distinguish data frame from field. If you are routinely working with one or a few data frames, R can be told the name(s) of the data frames in an "attach" statement, and the data frame name and separator can be omitted. For example, if we give the command

```
attach(maha)
```

we can specify the field "Elev" simply as "Elev" rather than "maha$Elev." This is more concise notation, but means that we cannot have a variable with the same name as a field in a data frame that is attached. Data frames are extraordinarily useful in R.

Lists
As noted above, a list is a compound object composed of associated data. Items within a list are generally referred to as components. Similar to data frames, components in a list can be given a name, and the component can be specified by name at any time. In addition, components can be specified by their position in the list, similar to a subscript in a vector. However, in contrast to a vector, lists components are specified in double [[ ]] delimiters. We will ultimately find it quite handy to create our own lists, but for the first few labs we will just see them as results from analyses, so we'll take them as they come and demonstrate their properties by example.

R VECTOR AND MATRIC OPERATORS

Because R is a 4th generation language, it is often possible to perform fairly sophisticated routines with little programming. The key is to recognize that R operates best on vectors, matrices, or data fames, and to capitalize on that. A large number of functions exist for manipulating vectors, and by extension, matrices. For example, if veg is a plant community matrix of 100 sample sites and 200 species (sites as rows and species as columns), we can perform the following:

`x <- max(veg[,3])` --- assigns the maximum value of species 3 among all sites
`y <- sum(veg[,5])` --- assigns the sum of species 5 abundance in all sites to y
`logveg <- log(veg+1)` --- creates a new matrix called "logveg" with all values the log of the respective values in veg (+1 to avoid log(0) which is undefined)

In addition, R supports logical subscripts, where the subscript is applied whenever the logical function is true. Logical operators include:

> for "greater than"
>= for "greater than or equal to"
< for "less than"
<= for "less than or equal to"
== for "equal to"
!= for "not equal to"
& for "and"
| for "or"

For example:

`q <- sum(veg[,8]>10)` --- assigns q the number of sites where the abundance of species 8 is greater than 10 (veg[,8]>0 is evaluated as 1 (true) or 0 (false), so that the sum is of 0's and 1's).
`r <- sum(veg[,8][veg[,8]>10])` --- assigns r the sum of the abundance for species 8 in sites where species 8 has abundance greater than 10
`deep14 <- max(veg[,14][soil=='deep'])` --- assigns the maximum abundance for species 14 on sites with deep soils

A final case is of special note. Missing values in a vector or matrix are always a problem in ecological data sets. Sometimes it is best simply to remove samples with missing data, but often only one or a few values are missing, and it's best to keep the sample in the matrix with a suitable missing value code. For now lets assume that we have missing values in a vector. To use all of the vector EXCEPT the missing value, use

```
spc.site[!is.na(spc.site)]
```

That's complicated enough to merit some discussion. The R function to identify a missing value is

```
is.na( )
```

so that to say all of a vector except missing values, we set a logical test to be true when values are not missing. Since the R operator for "not" is !, the correct test is

```
!is.na( )
```

and to specify which vector we're testing for missing value, we put the vector in parentheses as follows:

```
!is.na(spc.site)
```

Accordingly, the full expression is

```
spc.site[!is.na(spc.site)]
```

This use of missing values is critical to R because all operations on vectors or matrices must have the same number of elements. So, if there are missing values in any field we're using in a calculation, the same record (row) must be omitted from all the other fields as well.

ROW OR COLUMN OPERATIONS ON A MATRIX

Vector operators can be applied to every row or column of a matrix to produce a vector with the apply command. For example:

`spcmax <- apply(veg,2,max)` creates a vector "spcmax" with the maximum value for each species in its respective position. The apply operator is employed as:

```
apply("matrix name",1(rowwise) or 2(columnwise),vector operator)
```

so that

```
sitesum <- apply(veg,1,sum)
```

creates a vector of total species abundance in each site. The vector is as long as the number of rows in matrix veg.

TRIANGULAR MATRICES

Often in community ecology we work with symmetric matrices (e.g. similarity, dissimilarity, of distance matrices). These matrices take up extra space (since the value of the diagonal is known by definition, and since every other value is stored twice (matrix[x,y]=matrix[y,x]). We can save space by only storing one triangle of the matrix. In addition, some analyses require a vector argument, rather than a matrix, and it's convenient to convert the triangular matrix to a vector. This can be done as follows:

```
triang <- matrix[row(matrix) > col(matrix)]
```

GRAPHING IN R

R has a powerful graphics capability that is much of the appeal to using the system. Many of the analyses have special plotting capabilities that allow you to plot results without storing multiple intermediate products. R supports a fairly broad range of graphic devices in addition to excellent on-screen plotting.

You will want to send some time exploring the Cookbook for R (http://www.cookbook-r.com/). This resource includes the R Graphics Cookbook that provides detailed instructions on how to visual your data using the ggplot2 package (https://ggplot2.tidyverse.org/). I also recommend using the Quick-R reference (http://www.statmethods.net/). See Canvas website for additional resources.

SOME COMMON PLOTTING FUNCTIONS

(borrowed from *Statistics using R with Biological Examples*, Seefeld and Linder)

Function name	**Plot produced**
boxplot(x)	"Box and whiskers" plot
pie(x)	Circular pie chart
hist(x)	Histogram of the frequencies of x
barplot(x)	Histogram of the values of x
stripchart(x)	Plots values of x along a line
dotchart(x)	Cleveland dot plot
pairs(x)	For a matrix x, plots all bivariate pairs
plot.ts(x)	Plot of x with respect to time (index values of the vector unless specified)
contour(x,y,z)	Contour plot of vectors x and y, z must be a matrix of dimension rows=x and columns=y
image(x,y,z)	Same as contour plot but uses colors instead of lines
persp(x,y,z)	3-d contour plot

HIGH-LEVEL PLOTTING
(borrowed from *R for Beginners*, Emmanuel Paradis)

`plot(x)`	plot of the values of `x` (on the y-axis) ordered on the x-axis
`plot(x, y)`	bivariate plot of `x` (on the x-axis) and `y` (on the y-axis)
`sunflowerplot(x, y)`	id. but the points with similar coordinates are drawn as a flower which petal number represents the number of points
`pie(x)`	circular pie-chart
`boxplot(x)`	"box-and-whiskers" plot
`stripchart(x)`	plot of the values of `x` on a line (an alternative to `boxplot()` for small sample sizes)
`coplot(x~y | z)`	bivariate plot of `x` and `y` for each value (or interval of values) of `z`
`interaction.plot (f1, f2, y)`	if `f1` and `f2` are factors, plots the means of `y` (on the y-axis) with respect to the values of `f1` (on the x-axis) and of `f2` (different curves); the option `fun` allows to choose the summary statistic of `y` (by default `fun=mean`)
`matplot(x,y)`	bivariate plot of the first column of `x` *vs.* the first one of `y`, the second one of `x` *vs.* the second one of `y`, etc.
`dotchart(x)`	if `x` is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)
`fourfoldplot(x)`	visualizes, with quarters of circles, the association between two dichotomous variables for different populations (`x` must be an array with `dim=c(2, 2, k)`, or a matrix with `dim=c(2, 2)` if $k = 1$)
`assocplot(x)`	Cohen–Friendly graph showing the deviations from independence of rows and columns in a two dimensional contingency table
`mosaicplot(x)`	'mosaic' graph of the residuals from a log-linear regression of a contingency table
`pairs(x)`	if `x` is a matrix or a data frame, draws all possible bivariate plots between the columns of `x`
`plot.ts(x)`	if `x` is an object of class `"ts"`, plot of `x` with respect to time, `x` may be multivariate but the series must have the same frequency and dates
`ts.plot(x)`	id. but if `x` is multivariate the series may have different dates and must have the same frequency
`hist(x)`	histogram of the frequencies of `x`
`barplot(x)`	histogram of the values of `x`
`qqnorm(x)`	quantiles of `x` with respect to the values expected under a normal law
`qqplot(x, y)`	quantiles of `y` with respect to the quantiles of `x`
`contour(x, y, z)`	contour plot (data are interpolated to draw the curves), `x` and `y` must be vectors and `z` must be a matrix so that `dim(z)=c(length(x), length(y))` (`x` and `y` may be omitted)
`filled.contour (x, y, z)`	id. but the areas between the contours are coloured, and a legend of the colours is drawn as well
`image(x, y, z)`	id. but the actual data are represented with colours
`persp(x, y, z)`	id. but in perspective
`stars(x)`	if `x` is a matrix or a data frame, draws a graph with segments or a star where each row of `x` is represented by a star and the columns are the lengths of the segments

LOW-LEVEL PLOTTING

R has a set of graphical functions that affect an already existing graph: they are called low-level plotting commands. Here are the main ones (borrowed from *R for Beginners*, Emmanuel Paradis):

`points(x, y)`	adds points (the option `type=` can be used)
`lines(x, y)`	id. but with lines
`text(x, y, labels, ...)`	adds text given by `labels` at coordinates (`x`,`y`); a typical use is: `plot(x, y, type="n"); text(x, y, names)`
`mtext(text, side=3, line=0, ...)`	adds text given by `text` in the margin specified by `side` (see `axis()` below); `line` specifies the line from the plotting area
`segments(x0, y0, x1, y1)`	draws lines from points (`x0`,`y0`) to points (`x1`,`y1`)
`arrows(x0, y0, x1, y1, angle= 30, code=2)`	id. with arrows at points (`x0`,`y0`) if `code=2`, at points (`x1`,`y1`) if `code=1`, or both if `code=3`; `angle` controls the angle from the shaft of the arrow to the edge of the arrow head
`abline(a,b)`	draws a line of slope `b` and intercept `a`
`abline(h=y)`	draws a horizontal line at ordinate `y`
`abline(v=x)`	draws a vertical line at abcissa `x`
`abline(lm.obj)`	draws the regression line given by `lm.obj` (see section 5)
`rect(x1, y1, x2, y2)`	draws a rectangle which left, right, bottom, and top limits are `x1`, `x2`, `y1`, and `y2`, respectively
`polygon(x, y)`	draws a polygon linking the points with coordinates given by `x` and `y`
`legend(x, y, legend)`	adds the legend at the point (`x`,`y`) with the symbols given by `legend`
`title()`	adds a title and optionally a sub-title
`axis(side, vect)`	adds an axis at the bottom (`side=1`), on the left (`2`), at the top (`3`), or on the right (`4`); `vect` (optional) gives the abcissa (or ordinates) where tick-marks are drawn
`box()`	adds a box around the current plot
`rug(x)`	draws the data `x` on the x-axis as small vertical lines
`locator(n, type="n", ...)`	returns the coordinates (x, y) after the user has clicked `n` times on the plot with the mouse; also draws symbols (`type="p"`) or lines (`type="l"`) with respect to optional graphic parameters (`...`); by default nothing is drawn (`type="n"`)

GRAPHICAL PARAMETERS

In addition to low-level plotting commands, the presentation of graphics can be improved with graphical parameters. They can be used either as options of graphic functions (but it does not work for all), or with the function **`par`** to change permanently the graphical parameters, i.e. the subsequent plots will be drawn with respect to the parameters specified by the user. For instance, the following command:

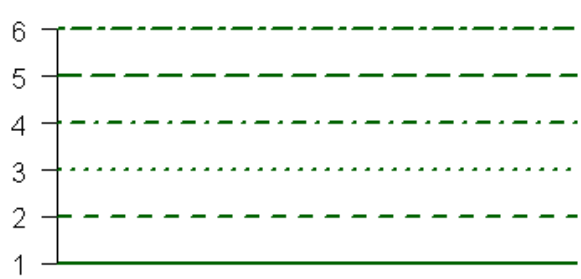
```
par(bg="yellow")
```

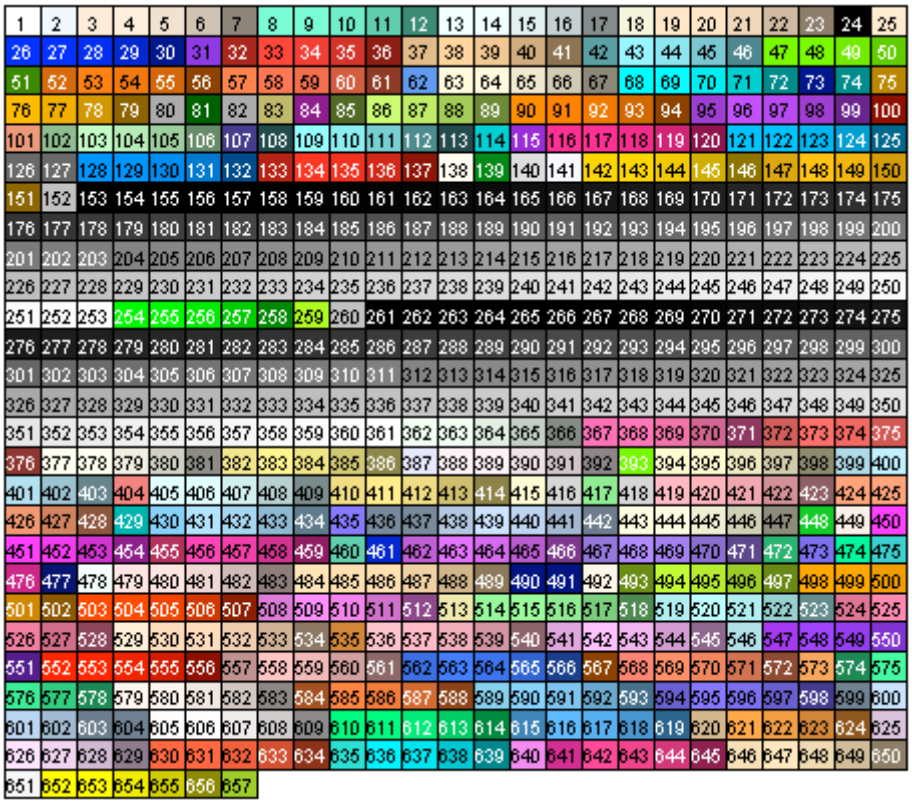
will result in all subsequent plots drawn with a yellow background. There are 73 graphical parameters, some of them have very similar functions. The exhaustive list of these parameters can be read with **`?par`**; I will limit the following table to the most usual ones (borrowed from *R for Beginners*, Emmanuel Paradis).

`adj`	controls text justification with respect to the left border of the text so that `0` is left-justified, `0.5` is centred, `1` is right-justified, values > 1 move the text further to the left, and negative values further to the right; if two values are given (e.g., `c(0, 0)`) the second one controls vertical justification with respect to the text baseline
`bg`	specifies the colour of the background (e.g., `bg="red"`, `bg="blue"`; the list of the 657 available colours is displayed with `colors()`)
`bty`	controls the type of box drawn around the plot, allowed values are: `"o"`, `"l"`, `"7"`, `"c"`, `"u"` ou `"]"` (the box looks like the corresponding character); if `bty="n"` the box is not drawn
`cex`	a value controlling the size of texts and symbols with respect to the default; the following parameters have the same control for numbers on the axes, `cex.axis`, the axis labels, `cex.lab`, the title, `cex.main`, and the sub-title, `cex.sub`
`col`	controls the colour of symbols; as for `cex` there are: `col.axis`, `col.lab`, `col.main`, `col.sub`
`font`	an integer which controls the style of text (`1`: normal, `2`: italics, `3`: bold, `4`: bold italics); as for `cex` there are: `font.axis`, `font.lab`, `font.main`, `font.sub`
`las`	an integer which controls the orientation of the axis labels (`0`: parallel to the axes, `1`: horizontal, `2`: perpendicular to the axes, `3`: vertical)
`lty`	controls the type of lines, can be an integer (`1`: solid, `2`: dashed, `3`: dotted, `4`: dotdash, `5`: longdash, `6`: twodash), or a string of up to eight characters (between `"0"` and `"9"`) which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example `lty="44"` will have the same effet than `lty=2`
`lwd`	a numeric which controls the width of lines
`mar`	a vector of 4 numeric values which control the space between the axes and the border of the graph of the form `c(bottom, left, top, right)`, the default values are `c(5.1, 4.1, 4.1, 2.1)`
`mfcol`	a vector of the form `c(nr,nc)` which partitions the graphic window as a matrix of `nr` lines and `nc` columns, the plots are then drawn in columns (see section 4.1.2)
`mfrow`	id. but the plots are then drawn in line (see section 4.1.2)
`pch`	controls the type of symbol, either an integer between 1 and 25, or any single character within `""` (Fig. 2)
`ps`	an integer which controls the size in points of texts and symbols

PLOTTING SYMBOLS, LINES AND COLORS

Below are the plotting symbols in R (pch=1:25). The colors were obtained with the options **`col="blue"`**, **`bg="yellow"`**, the second option has an effect only for the symbols 21-25. Any character can be used (pch="*", "?", ".", . . .). Borrowed from *R for Beginners* by Emmanuel Paradis.





SAVING GRAPHICS

Notice that when the graphics window is active the main menu is different. On the File menu there are many options for saving a graphic, including different graphical formats (png, bmp, jpg) and other formats (metafile, postscript, pdf). You could also use command line functionality to save, but using the options under the File menu is easier and pops up a save as dialog box allowing you to choose the directory you are saving the graphic file to.

Another option to save a graphic is to simply right mouse click on the graphic, which will produce a pop up menu with options to copy or save the graphic in various formats as well as to directly print the graphic. In a Windows environment the copy options are as a bitmap or metafile, and the save options are as metafile or postscript.

SUMMARY

In this document, we have looked as some of the strengths that make R an attractive option for students, researchers, statistician, and data analysts trying to understand the meaning of their data. We have walked through the program's installation and talked about how to enhance R's capabilities by downloading additional packages. We have explored the basic interface, running programs interactively and in batch, and produced a few sample graphs. We have also learned how to save our work to both text and graphic files. Hopefully, you're getting a sense of how powerful this freely available software can be. Now that we have R up and running, it's time to get our data into the mix.

ADDITIONAL RESOURCES

CRAN	http://www.r-project.org/
Quick-R	http://www.statmethods.net/
Graphing in R	http://www.statmethods.net/graphs/creating.html

GETTING HELP

Virtually everything in R has some type of accessible help documentation. The challenge is finding the documentation that answers your question. This section gives some suggestions for where to look for help.

Program Help Files: The quickest and most accessible source of help when using R is to use the on-line help system that is part of R. This includes on-line documentation for the R base packages, as well as on-line documentation for any loaded packages. Finding help when you know the name of what it is you're asking for help on is easy, just use the help function with the topic of interest as the argument of the help function. For example to get help on function sum:

```
help(sum)
```

As an alternative to calling the help function you can just put a question mark in front of the topic of interest:

```
?sum
```

Note that on-line help is especially useful for describing function parameters, which this document will not discuss in great detail because many functions in R have lots of optional parameters (read.table for example has about 12 different optional parameters). You should become comfortable looking up functions using help to get detailed parameter information for use in R.

2. Data Screening for Multivariate Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Checking for data irregularities
- Transforming and standardizing data

R Packages: vegan, pastecs
R Source: biostats.R

BACKGROUND

The development of an appropriate multivariate analytical strategy for a given data set should proceed as a careful sequence of steps, in which results obtained at a given step determine subsequent ones (Kenkel 2006). Multivariate data analysis is thus a process of adaptive learning, in which decisions made at a given analytical stage influence subsequent steps and strategies. Before proceeding with a formal multivariate analysis, it is important to complete a detailed exploratory analysis of the data. Exploratory analysis is undertaken to elucidate and summarize distributional properties and underlying trends of the data, which in turn direct the user to meaningful analyses and interpretations. In other words, data screening is an essential precursor to all multivariate analyses. Below we discuss how to search and correct data errors, treat missing data, detect and handle insufficiently sampled variables, conduct transformations and standardizations, and detect and handle outliers. This document is based, in part, on the functions (and help files) available in biostats.R written by Kevin McGarigal at the University of Massachusetts.

SET-UP

First, set-up your R work session by setting the current work directory to your folder of choice and load the vegan and pastecs Packages and source the biostats.R file from the *File* pull-down menu in R ("Source R code…") or the *Edit* pull-down menu in RStudio ("Source file"). You can also do this using the functions **`setwd`**, **`library`** and **`source`** (see Chapter 1: Introduction to R) directly in R console, or selecting the appropriate box in RStudio. Note that you might need to install the package in RStudio if it does not appear in the window.

Second, open a new R script window by selecting the *File* pull-down menu in RStudio, and click on *New* and then *R Script*. You will be typing all your R commands in this window. Immediately save your R script (and don't forget to save repeatedly during every exercise). You can create separate R scripts for every R tutorial for build a master R script containing all the functions.

Third, import the MAHA environment dataset and the MAHA species abundance dataset. You can do this directly using the .csv files or practice by preparing the data in Excel and save the species abundance data and environment data into separate comma delimited (.csv) files (see Chapter 1). Do not import data using the "Import Dataset" option in RStudio because you will not be able to assign the first column as row names. Although you can always do this after the fact. Instead, you can import the data (according to the file names selected below) by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
spedata <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

We can examine the structure of the data set by typing:

```
str(envdata)
```

CALCULATE SUMMARY STATISTICS AND SCREEN FOR DATA ERRORS

Let's explore the data and screen for errors by calculating a series of summary statistics. In particular, note the minimum and maximum values for obvious errors, and examine the summary statistics to gain a general understanding of the range of values observed and the frequency of zeros and missing values, as well as coefficient of variation.

Using the stat.desc function in pastecs library, let's type:

```
stat.desc(envdata)
```

This will return the following:

```
               Sinuosity      Slope     WDRatio     SubEmbed     HabQual
Elev
nbr.val      45.00000000 45.0000000  45.0000000   45.0000000  45.0000000
nbr.null      0.00000000  0.0000000   0.0000000    0.0000000   0.0000000
nbr.na        0.00000000  0.0000000   0.0000000    0.0000000   0.0000000
min           1.00723514  0.1550000   8.1542127   10.6122449   7.0800000
max           2.00679467  9.1000000  53.0179123   96.9090909  18.8300000
range         0.99955953  8.9450000  44.8636996   86.2968460  11.7500000
sum          51.81200587 68.5620000 935.0573022 2577.3141256 611.0000000
median        1.09379217  1.1500000  20.0346580   60.1851852  13.5000000
mean          1.15137791  1.5236000  20.7790512   57.2736472  13.5777778
. . .
```

To produce a column summary (i.e. species), type:

```
stat.desc(spedata)
```

In this output, the field "nbr.null" represents the number of objects (sites) in which the variable (species abundance) equals zero.

There are numerous other R Packages that also provide functions to explore your data. These include:

library(Hmisc) – describe()
library (psych) – describe(), describeBy()
library(skimr) – skim(), skim_with()
library(summarytools) – descr(), dfSummary()

EVALUATE MISSING DATA AND TAKE CORRECTIVE ACTION

Review the column and row summary statistics above and note the amount and pattern of missing data. If missing data values exist and are sparsely distributed throughout the data frame, consider eliminating data with missing data or replacing the small amounts of missing values with either the median and mean of the column or using statistical interpolation. For count variables (e.g., species abundance data), the median is perhaps the more logical replacement, as it maintains the integer status of all values. To accomplish a 'median' replacement, type:

```
testdata <- replace.missing(spedata)
```

Note that testdata is identical to spedata because the original data contains no missing values.

Alternatively, with other types of numeric data the column mean may be considered a logical replacement as well. To accomplish a 'mean' replacement, type:

```
testdata <- replace.missing(spedata,method='mean')
```

If missing data values exist and are concentrated in one or more samples and/or one or more variables, consider dropping the offending samples and/or variables. For example, consider dropping variables with too many missing values, say >5 percent of observations missing, using the **`drop.var()`** function, by typing:

```
testdata <- drop.var(spedata, pct.missing=5)
```

Again, note that testdata is identical to spedata because the original data contains no missing values. What a wonderful dataset!

EXAMINE FREQUENCY OF OCCURRENCE AND ABUNDANCE GRAPHS AS APPROPRIATE

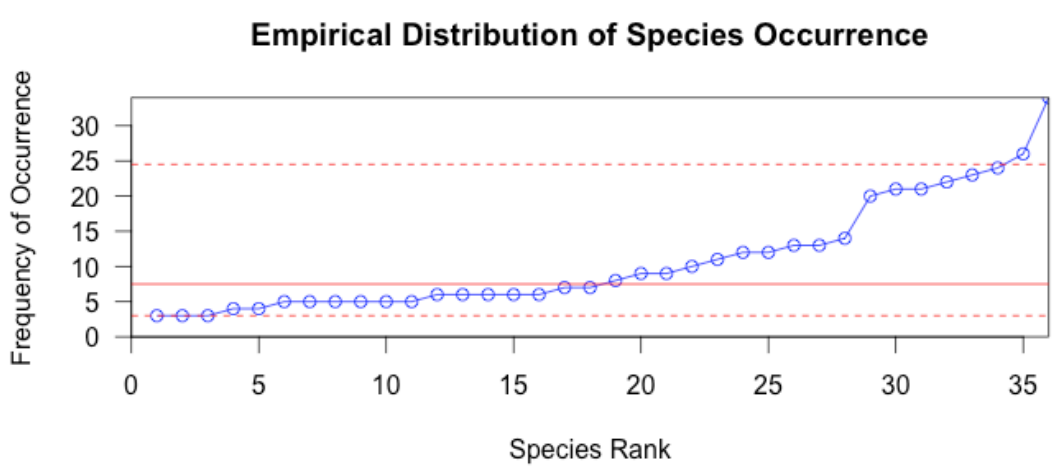
If you are analyzing community data (such as the MAHA data set) that contain samples by species abundances or occurrence, there are several things about the species and samples you may wish to know:

1. In how many sites does each species occur?
2. What is the mean abundance of each species when it occurs (not averaging zeros for sites where it is absent)?
3. Is the mean abundance of species correlated with the number of sites in which it occurs?
4. How many species occur in each site?
5. Is the total abundance of species in a site correlated with the number of species in a site?

`Foa.plots()` produces a series of 10 different plots that can help you answer these questions and more. To view the plots for the MAHA fish abundance data set (spedata), type:

```
foa.plots(spedata)
```

This will return the following in the Graphics window. Press return to toggle through the graphs.



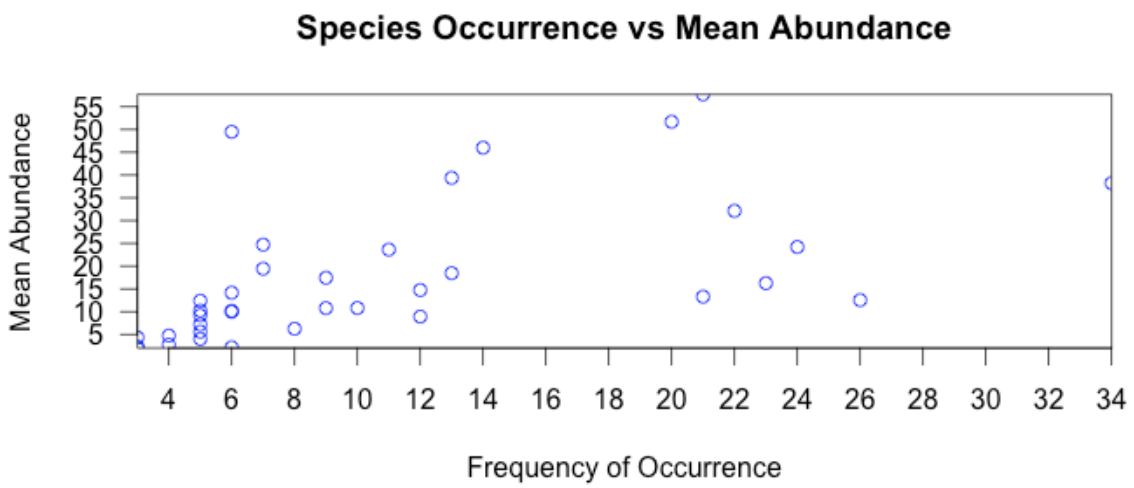
In how many sites does each species occur? The first four plots portray the species' frequency of occurrence among plots in a number of different ways – either as an empirical cumulative distribution function (ECDF) of species occurrence or as a histogram of species occurrence. Because some ecologists have suggested that species occurrence and abundance distributions sometimes follow a log-normal distribution, the fourth plot depicts a histogram of log-transformed species occurrence.

What is the mean abundance of each species when it occurs (not averaging zeros for sites where it is absent)? The fifth plot is an ECDF of species mean abundance.

Is the mean abundance of species correlated with the number of plots they occur in? The sixth plot is a scatter plot of frequency of occurrence against mean abundance. Is there any apparent relationship between the two? Are the widespread species also generally more abundant? Are there many widespread species that occur at low abundance? Conversely, are there species much less widespread, but abundant where they occur?

To see which dot is which species, answer 'y' to the question on the console and then simply click on the point and the species acronym (variable name) will appear next to the point. When you are done identifying points, simply right click "Stop" or press "Esc".

This will return the following in the Graphics window:



As before, it may be more meaningful to look at the log of mean abundance, as often abundance follows a log-normal distribution. The seventh plot is the same as above except that mean abundance has been log-transformed. Are the patterns the same?

Is the total abundance of species in a site correlated with the number of species in a site? To answer this question, first it is instructive to look at the number of species per site. The eighth plot depicts the ECDF of site richness. Are there any interesting patterns? For example, do most sites support an average number of species, while only a few sites supporting either very few or very many species? Or is the pattern different?

Second, what is the pattern in the distribution of site total abundance? The ninth plot is the ECDF of total site abundance. How does it compare to the ECDF of site richness?

Finally, to answer the question on the relation between total abundance and number of species/site, the last plot is a scatter plot of the two variables. Is there is relationship between the number of

species per site and the total abundance? Do species-rich sites generally support a greater total abundance of species as well?

EXAMINE VARIABLES FOR "SUFFICIENCY" AND ELIMINATE IF NECESSARY

It is important to screen your data for insufficient variables (i.e., rare species, abundant species, variables with low variation: see lecture notes for a detailed discussion).

Review the column summary statistics above and note the number and percentage of zeros and the coefficient of variation of each variable. Consider dropping rare species having non-zero values in less than 5% of the sites, by typing:

```
testdata <- drop.var(spedata,min.po=5)
```

This should result in no change in the dataset (type **`str(testdata)`** to examine this).

Now, consider dropping variables with fewer than 5 non-zero values (i.e., sites), by typing:

```
testdata <- drop.var(spedata,min.fo=5)
str(spedata)
str(testdata)
```

The following will be returned and will indicate that 5 species have been deleted from the data set.

```
'data.frame':   45 obs. of  36 variables:
 $ BANDDART: int  0 0 0 0 0 0 0 0 5 0 ...
 $ BANDSCUL: int  0 0 0 0 0 2 0 0 0 0 ...
 $ BLACDACE: int  7 1 16 114 15 1 73 97 0 0 ...
. . .

'data.frame':   45 obs. of  31 variables:
 $ BANDSCUL: int  0 0 0 0 0 2 0 0 0 0 ...
 $ BLACDACE: int  7 1 16 114 15 1 73 97 0 0 ...
 $ BLUECHUB: int  372 91 1 0 3 0 0 0 0 0 ...
. . .
```

In community data sets, consider dropping abundant generalist species, for example those species occurring in more than 95% of the plots, by typing:

```
testdata <- drop.var(spedata,max.po=95)
```

Depending on the subsequent analysis, dropping abundant generalist species may or may not be desirable, so this decision should be made carefully.

Consider dropping variables with too little variation, for example those with cv<5, by typing:

```
testdata <- drop.var(spedata,min.cv=5)
```

Note: dropping variables with too little variation is unlikely to affect subsequent analyses, because they will not exert much influence over the data cloud.

EXAMINE THE DISTRIBUTIONAL PROPERTIES OF THE DATA

It is important to examine the distribution of each variable independently and, among other things, note the need for transformations to improve characteristics of the distribution (e.g., normality). Of course, there are many ways to examine distributions and we will not attempt an exhaustive review. Instead, we will focus on four common graphical approaches suitable for continuous variables. Note: categorical variables (or factors) are not suitable for these functions and will produce an error if they are not first removed from the input data set.

Empirical cumulative distribution functions: The empirical cumulative distribution function (ECDF) is a simple rank order distribution of increasing values of the variable. A variable with a perfectly uniform distribution of values within its range (minimum to maximum), so that no one value is more common than another, will have points that fall on a perfect diagonal straight line. Deviations from the diagonal indicate non-uniformity. Plot the ECDF for each species (in turn) by typing:

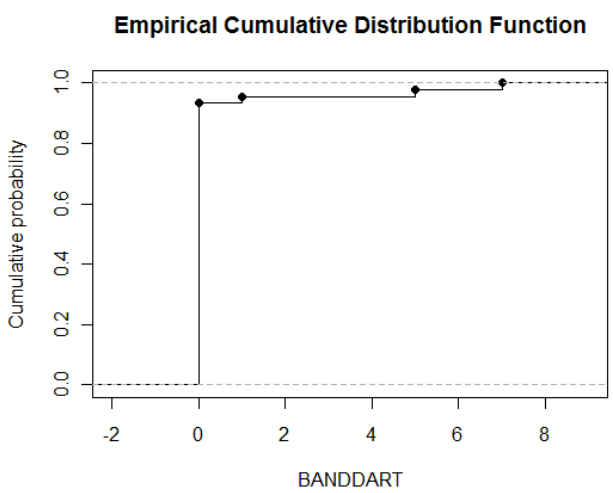
```
ecdf.plots(spedata)
```

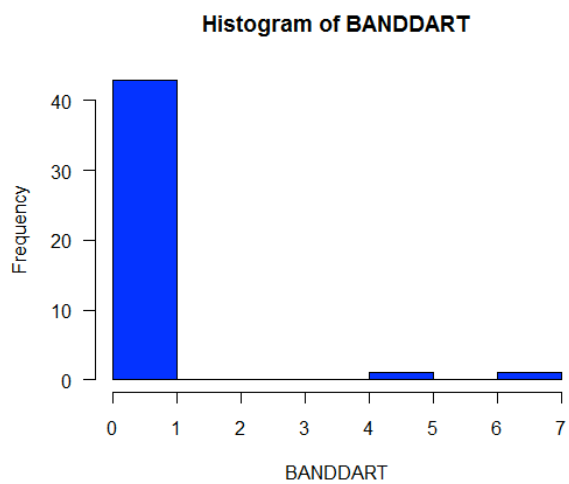
Histograms: Next, try the more conventional histogram, by typing:

```
hist.plots(spedata)
```

Ecological data are often highly skewed, so evaluate each distribution for skewness. You should also pay attention to the occurrence of extreme values in the distribution (i.e., potential outliers), as we will address this issue below.

The following two plots will be returned separately for the first species in the dataset (banded darter):





Box-and-whisker plots: An alternative way to examine the distribution of each variable is with a box-and-whisker plot, obtained by typing:

```
box.plots(spedata)
```

Box plots can depict the skewness of the distribution quite nicely and also can be used to identify extreme observations. The central box shows the data between the 'hinges' (roughly quartiles), with the median represented by a line. 'Whiskers' go out to the extremes of the data, and very extreme points (defined as samples that are farther than 1.5 (by default) times the inter-quartile range from the box) are shown by themselves.

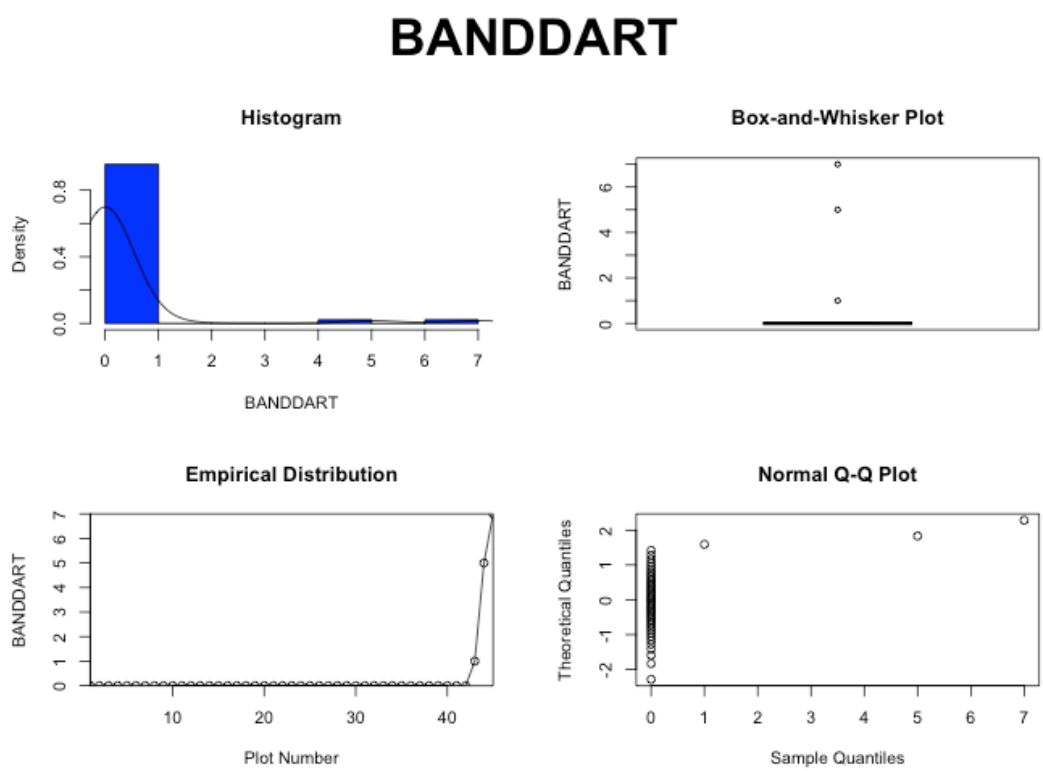
Normal quantile-quantile plots: Another useful way of examining the distribution of each variable is to compare the empirical cumulative distribution function (ECDF) to the expected ECDF for a normal distribution. A normal quantile-quantile (or QQ) plot does just this. Try typing:

```
qqnorm.plots(spedata)
```

Note: the qqnorm plot depicts the sample quantiles on the x axis against the theoretical quantiles from a normal distribution of the same sample size on the y axis. If the data are from a perfectly normal distribution, the data will lie on a diagonal straight line. Departures from the diagonal indicate deviations from a normal distribution. Skewed distributions show up nicely as deviations from the line at the tails.

Four-in-one plots: The four plots described above can be depicted together in a single 4-part plot for each variable. The four-in-one plot is generated by typing:

```
uv.plots(spedata)
```



Hot damn!

EVALUATE THE NEED FOR DATA TRANSFORMATION

Once you have thoroughly examined your data, you may deem it necessary or useful to transform your data. Ecological data are commonly skewed and/or range over several orders of magnitude, and as such can benefit from a transformation, such as the log or square-root transformations that compress large values. For community data sets involving species abundances, it is sometimes useful or more meaningful to transform the data to binary (presence/absence) data. Use the **`data.trans()`** function to transform variables using the log, power or arcsine square-root transformations.

Log (base 10) transformation: To log-transform the species abundances in the MAHA fish data set (spedata), type:

```
data.trans(spedata,method='log')
```

Examine the paired histograms comparing the raw (untransformed) and transformed distributions to see the effect of the transformation.

Power transformation (including binary transformation): To power-transform the species abundance data, try a squareroot transformation, by typing:

```
data.trans(spedata,method='power',exp=.5)
```

Note that the squareroot transformation is simply a special case of the power transformation where the exponent is equal to 0.5.

To transform the species abundances into presence/absence (i.e., binary transformation), use the power method with an exponent equal to zero, by typing:

```
data.trans(spedata,method='power',exp=0)
```

Arcsine square-root transformation: For proportional data (i.e., ranges 0-1), the arcsine square-root transformation is often recommended by statisticians. To accomplish this, type:

```
data.trans(spedata,method='asin')
```

The **`decostand()`** function in the vegan library is also very useful and will be described below.

Its usage is:

decostand(x, method, MARGIN, range.global, logbase = 2, na.rm=FALSE, ...)

The relevant arguments are:

x – the data matrix to be analyzed
method – Standardization method. See Details for available options.
margin - Margin, if default is not acceptable. 1 = rows, and 2 = columns of x
etc …

The function offers following standardization methods for community data:

total: divide by margin total (default MARGIN = 1).
max: divide by margin maximum (default MARGIN = 2).
freq: divide by margin maximum and multiply by the number of non-zero items, so that the average of non-zero entries is one (default MARGIN = 2).
normalize: make margin sum of squares equal to one (default MARGIN = 1).

range: standardize values into range 0 ... 1 (default MARGIN = 2). If all values are constant, they will be transformed to 0.
standardize: scale x to zero mean and unit variance (default MARGIN = 2).
pa: scale x to presence/absence scale (0/1).
chi.square: divide by row sums and square root of column sums, and adjust for square root of matrix total (Legendre & Gallagher 2001).
hellinger: square root of method = "total" (Legendre & Gallagher 2001).
log: logarithmic transformation as suggested by Anderson et al. (2006): *log_b (x) + 1* for *x > 0*, where *b* is the base of the logarithm; zeros are left as zeros. This is not the *log(x+1)* transformation,

For additional documentation type **`?decostand`** at the R prompt.

Note: for any of these transformations, it is common to apply the same transformation to an entire set of related variables so that they are all on the same scale. For example, if some of the species variables benefit from a log transformation, it would generally be preferably to log-transform all species variables, not just the ones in need.

Once you have decided to use a particular transformation, you will need to save the transformed data into a new object by assigning the results, or you can use the 'outfile' argument in the **`data.trans()`** function to automatically save the transformed data set to a new permanent data set. You could also take a simple approach by using one of the many standard operations available in R. For example, if you decide on a natural log transformation, then type:

```
tradata <- data.trans(spedata,method='log',plot=F) or

tradata <- log10(spedata+1)
```

You will note that we included "+1" because the matrix contains many zeros and therefore performing a simple log transformation is not possible. The **`data.trans`** function automatically adds "1" to all cells to account for zero values.

EVALUATE THE NEED FOR DATA STANDARDIZATION

In many ecological data sets, especially community data sets involving species abundances, it is often quite useful to standardize (or relativize) the data before conducting subsequent analyses. Data "standardization" involves adjusting a data value relative to a specified standard derived from the corresponding row (sample) or column (variable) of the data frame. Keep in mind that standardizations can fundamentally alter the patterns in the data and can make the difference "between illusion and insight, fog and clarity" (McCune and Grace 2002).

If you are working with community data involving species abundances, such as in the MAHA data set, check the coefficient of variation (cv) in row and column totals for the species variables (see **`stat.desc()`** described previously). For the column summary, this is the cv in species' total abundances (i.e. the cv in column totals). The corresponding value in the row summary is the cv in site totals (i.e. the cv in row totals). If these values are small, say <50, it is unlikely that standardization will accomplish much. However, if these values are large, say >100, then it is likely that standardization will have a large effect on the results.

If you are working with environmental data involving variables measured in different units or on different scales, consider column standardization (by norm or standard deviates).

There are many possible row and column standardizations and they will not be reviewed here. Refer to the BIOSTATS help file for **`data.stand()`** for details on the available standardizations. Before applying any standardization, be sure to understand what the standardization does and whether the

units for all the included variables are the same. Note: in some cases, standardization is built into the subsequent analyses and therefore unnecessary – but to know this requires that you already understand the mechanics of the techniques you intend to use (which we haven't gotten to yet!).

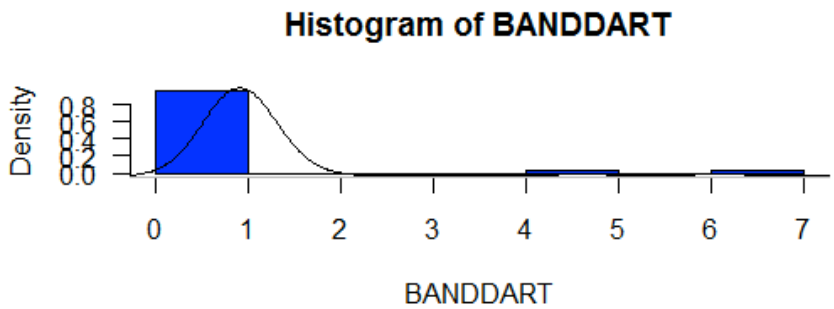
At this point, you might simply explore what various standardizations do to the data so that you are ready and able to standardize your data as needed when you decide on a particular statistical procedure.

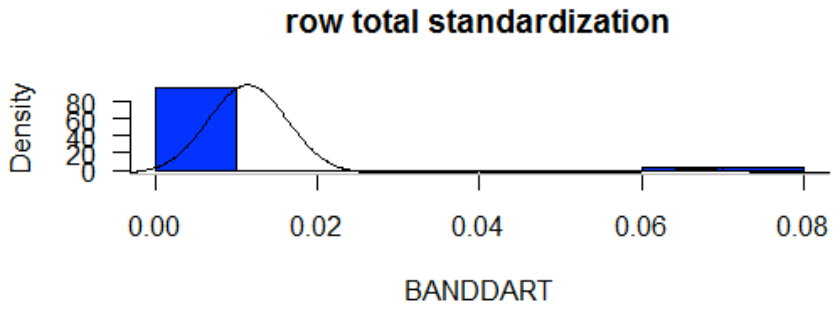
As an example, to conduct a 'row normalization' (i.e., to rescale each row so that the sums equal 1, or in other words to calculate relative species abundances), type the following:

```
data.stand(spedata,method='total',margin='row')
```

Examine the paired histograms that are produced by default to see the effect of the standardization on the distribution of each variable. Remember, if you intend to save the standardized results for use in a later analysis, you must save the results to a permanent data set. You can either assign the results to an object and then write that object to a file using the write.table() or write.csv() functions, or you can simply save the results directly with the data.stand() function by using the outfile='' argument.

Regardless of your decision to standardize or not, you should state your decision and justify briefly on ecological grounds.





EVALUATE THE DATA SET FOR OUTLIERS

Check your data set for potential outliers and eliminate them if justified. However, as a general rule, observations should not be dropped automatically just because they have extreme values. It is one thing to identify extreme observations that have high leverage on the results, but it is another thing altogether to delete these observations from the data set just because they have high leverage. What constitutes a true "outlier" depends on the question being asked and the analysis being conducted.

There is no general rule for deciding whether extreme observations should be considered "outliers" and deleted from the data set before proceeding with the analysis. Nevertheless, it is important to have an understanding of the number and pattern of extreme observations in order to gauge the robustness of the results. A good practice is to repeat the analyses with and without the suspect points and determine if the results are sensitive or robust to their inclusion. If the results are sensitive to the inclusion of these high-leverage points, you should probably carefully consider whether those points represent a meaningful ecological condition, and act on them accordingly.

There are several ways to identify extreme values, including both univariate and multivariate methods. It is always a good idea to begin with a univariate inspection. The uv.outliers() function computes the standardized values for each variable (i.e., z-scores) and returns a data frame with a list of samples and the variables that are greater than a specified number of standard deviations from the mean (default = 3). Let's do this for the MAHA environment data set. Try typing:

```
uv.outliers(envdata, id='Sinuosity:BasinAre', var='Elev', sd.limit=1)
```

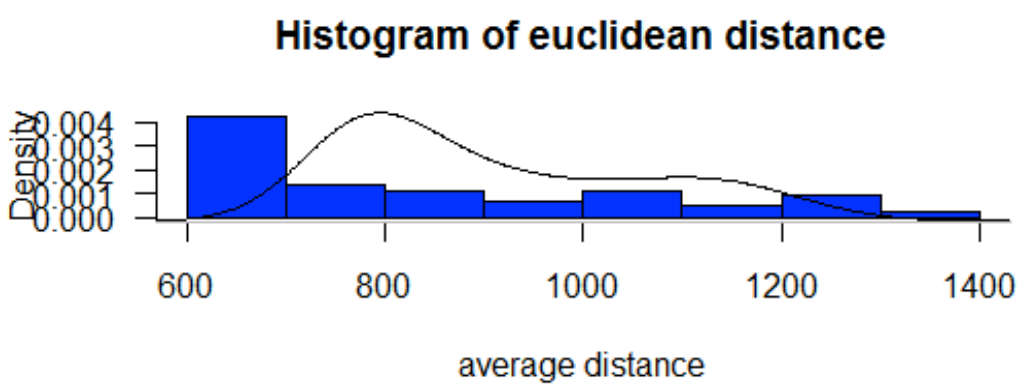
Note the number and distribution of extreme values based on 1 standard deviation. The default is sd.limit =3 based on the notion that 99.7% of all observations should fall within the mean +/- 3 SD according to a normal distribution. Are there observations with extreme values on several variables, or are the extreme values dispersed among samples and variables?

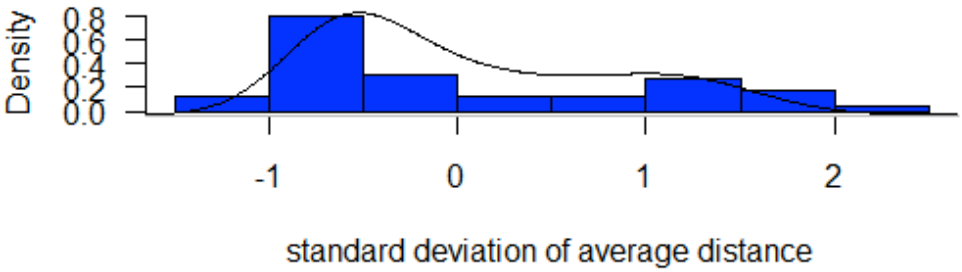
In the context of a multivariate data set, just because an observation is extreme on a single variable doesn't mean it is going to be a multivariate outlier. More importantly, an observation may not be a univariate outlier and yet still be an outlier when two or more variables are considered jointly. Thus, it is perhaps more instructive to determine if each observation is extreme in multivariate space.

To do this, use the **`multivar.outliers()`** function. Specifically, select an appropriate distance measure – ideally the distance measure to be used in subsequent analyses – and use this function to compute the average distance of each sample to all other samples. For example, for the MAHA environment data (envdata), try typing:

```
mv.outliers(envdata,method='euclidean', sd.limit=1)
```

This will produce a list of samples with an average Euclidean distance >1 standard deviations from the mean of average distances, in addition to paired histograms depicting the distribution of average distances and the distribution of standard deviates. Again, using the default of SD=3 is recommended (SD=1 is used here merely for illustrative purposes).





Another way of assessing multivariate outliers is to compute the Mahalanobis distance between each sample and the group of all other samples and to compare this against the expected distribution of Mahalanobis distances for a multivariate normal distribution, using a very conservative probability, e.g., p<0.001 based on a chi-square distribution with degrees of freedom equal to the number of variables. Try typing:

```
mv.outliers(envdata, method='mahalanobis', sd.limit=1)
```

OPTIONAL READINGS

Hagaman, R.M. and M.E. Morbeck. 1984. Data transformations in multivariate morphometric analyses. Journal of Human Evolution 13: 225-245

Hinch, S. G., and K. M. Somers. 1987. An experimental evaluation of the effect of data centering, data standardization, and outlying observations on principal components analysis. Coenoses 2:19–23.

Jackson, D. A. 1993. Multivariate analysis of benthic invertebrate communities: the implication of choosing particular data standardizations, measures of association, and ordination methods. Hydrobiologia 268:9–26.

Kenkel, NC 2006. On selecting an appropriate multivariate analysis. Canadian Journal of Plant Sciences 86: 663-676.

Legendre, P. and E.D. Gallagher. 2001. Ecologically meaningful transformations for ordination of species data. Oecologia 129: 271-280.

Noy-Meir, I. 1973. Data transformations in ecological ordination. I. Some advantages of non-centering. Journal of Ecology 61:329–341.

Noy-Meir, I., D. Walker, and W. T. Williams. 1975. Data transformations in ecological ordination. II. On the meaning of data standardization. Journal of Ecology 63:779–800.

Cao, Y., D.D. Williams, and N.E. Williams. 1999. Data transformation and standardization in the multivariate analysis of river water quality. Ecological Applications 9: 669-677.

EXERCISE

Purpose

One of the first steps in multivariate analysis is data screening using numerical and graphical approaches. Why? It (1) suggests a plausible model for the data, (2) assesses validity of model assumptions, (3) detects outliers, and (4) suggests possible data transformations and standardizations. Many multivariate methods assume that the data are multivariate normally distributed. It this exercise you will exploring your own data using the techniques described above.

Tasks

1. Produce summary statistics for each variable in your dataset(s).
    a. Are there missing values, and if so how do you plan on accounting for this problem?
2. Examine the distributional properties of your variables.
    a. Do outliers exist? What statistical and ecological criteria are you using to base your decision?
    b. Is there a need for data transformation? What transformations are you considering and why? Again, is your decision based on statistical or ecological criteria, or both?
    c. Is there is a need for data standardization? What standardizations are you considering and why?
3. If transformations and/or standardizations are appropriate, then produce histograms for each of the transformed variables (the same way you did for the raw data).
    a. How effective was your transformation/standardization?
    b. Will you use the raw or transformed/standardized data in subsequent analyses? Why?

3. Multivariate Resemblance

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Calculate similarities and dissimilarities among objects based on a set of descriptors

R Packages: vegan, simba, cluster, ecodist, gclus
R Source: biostats, coldiss

BACKGROUND

Ecologists have long collected quantitative observations to determine the resemblance between either objects under study or the variables describing them. Measuring the association between objects or descriptors is the first, and sometimes the only step in the numerical analysis of ecological data. The most common approach to assess resemblance is to first condense all (or the relevant part of) the information available in the data matrix into a square matrix of association among the objects or descriptors. In most instances the matrix is symmetric.

It follows that the structure resulting from the numerical analysis (i.e., the association matrix) does not necessarily reflect all the information originally contained in the ecological data matrix. This stresses the importance of choosing an appropriate measure of association! The choice determines the issue of the analysis. Therefore, the following facts must be considered:

- The nature of the study determines the structure of the data to be evaluated with an association matrix.
- The various measures available are subject to different mathematical constraints (depends upon whether one continues with ordination or clustering).
- Computational aspects such as what measures are available or can be programmed in particular software.

The analysis of similarities among objects (rows) is designated as Q mode analysis, whereas when relationships among variables (columns) are the focus of the study, this is referred to as R mode analysis (Legendre and Legendre 1998). Noticeably, the two modes of analysis do not generally use the same association coefficients. Although it is not possible to give a full review of all association coefficients here, it is useful to know that, for comparing objects (rows) based on their column attributes in Q mode analysis, coefficients may be chosen as a function of data type (quantitative, qualitative, ordinal, or mixed data, normalized data, presence/absence), importance given to rare species, weight given to each object, and calculation of associated probability levels.

In the following chapter, we will explore a large group of association coefficients based on the notion of similarity and dissimilarity. Throughout, the term “association coefficient” is used to describe any measure used to quantify the resemblance or difference between objects or descriptors.

SET-UP

First, set-up your R work session by setting the current work directory to your folder of choice and load the vegan, simba, ecodist, cluster Packages and source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Second, import the MAHA environment dataset, MAHA species abundance dataset and MAHA fish trait dataset by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
```

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
spetrait <- read.csv('MAHA_speciestrait.csv',header=TRUE, row.names=1)
```

Remember to screen your data for errors, outliers, missing values, etc … and conduct any data transformations or standardizations (see previous chapter). The MAHA dataset should look pretty good because it was pre-screened prior to being released to you. *Therefore, for the purposes of this computer lab you can use the raw environmental and species dataset*.

CALCULATING COEFFICIENTS OF SIMILARITY FOR BINARY DATA (SPECIES PRESENCE/ABSENCE)

Let's calculate the similarity among sites according to their fish community composition based on species presence/absence. First, let's transform the species abundances into presence/absence (i.e., binary transformation) using the power method with an exponent equal to zero, by typing:

```
speocc <- data.trans(speabu,method='power',exp=0,plot=F)
```

Remember to include **`plot=F`** so that you are not required to press enter for each plot. If you do not you must press enter for all plots before **`speocc`** is created. Alternatively you could directly import the data (assuming that you have prepared it already), by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
```

There are two very useful functions to compute similarity (**`sim`** function in the **`samba`** library) and dissimilar coefficients (**`vegdist`** function in the **`vegan`** library). I would highly recommend that you explore both!

Now, let's use the **`sim`** function in the **`simba`** library to calculate a series of (dis)similarity coefficients to describe the resemblance of sites according to their species composition. This library will compute 56 dis(similarity) measures for binary data, in this case species presence/absence.

Its usage is:

> sim(x, coord=NULL, method = "soer", dn=NULL, normalize = FALSE,
> listin = FALSE, listout = FALSE, ...)

The relevant arguments are:

1. x – the data matrix to be analyzed
2. method – Binary Similarity index including "soerensen", "jaccard", "ochiai", "mountford", "whittaker", "lande", "wilsonshmida", "cocogaston", "magurran", "harrison", "cody", "williams", "williams2", "harte", "simpson", "lennon", "weiher", "ruggiero", "lennon2", "rout1ledge", "rout2ledge", "rout3ledge", "sokal1", "dice", "kulcz1insky", "kulcz2insky", "mcconnagh", "manhattan", "simplematching", "margaleff", "pearson", "roger", "baroni", "dennis", "fossum", "gower", "legendre", "sokal2", "sokal3", "sokal4", "stiles", "yule", "michael", "hamann", "forbes", "chisquare", "peirce", "eyraud", "simpson2", "legendre2", "fager", "maarel", "lamont", "johnson", "sorgenfrei", "johnson2".
3. Etc …

For additional documentation type **`?sim`** at the R prompt.

Let's try calculating Jaccard's similarity matrix, by typing:

```
sp.jac <- sim(speocc, method = "jaccard")
```

Type **sp.jac**. The first upper left cells in the similarity matrix should be:

```
           S1         S2         S3         S4
S2  0.29411765
S3  0.61538462 0.46153846
S4  0.14285714 0.16666667 0.18181818
S5  0.56250000 0.27777778 0.46666667 0.30769231
```

Now calculate a series of different similarity matrices based on the Simple Matching coefficient and Sørenson's coefficient (note that Sørenson is incorrectly spelt in **sim** library … huh!). For example, you can type:

```
sp.sim <- sim(speocc, method = "simplematching")
sp.sor <- sim(speocc, method = " soerensen")
```

How do these matrices compare to each other? You should be able to note and <u>explain</u> the differences. We can do this graphically by plotting the pair-wise similarities based on two different coefficients against one another, by typing:

```
plot(sp.jac,sp.sim,xlab="Jaccard's coefficient",ylab="Simple Matching
coefficient")
```

We can add a 1:1 line to help with the comparisons, by typing:

```
abline(0,1,col="darkgray")
```

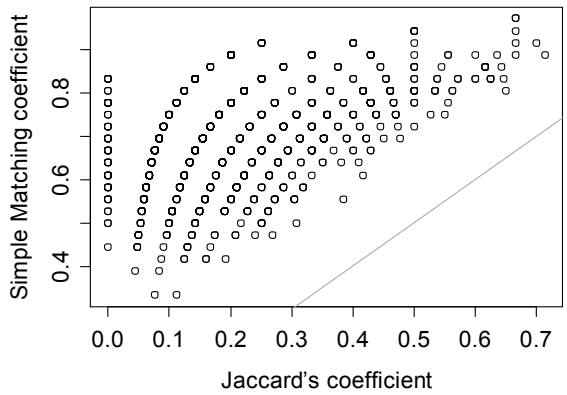
This should produce a plot resembling the one presented on the bottom left. Why do many sites exhibit 0% similarity based on Jaccard's coefficient but range between 45 and 85% similarity based on the Simple Matching coefficient?

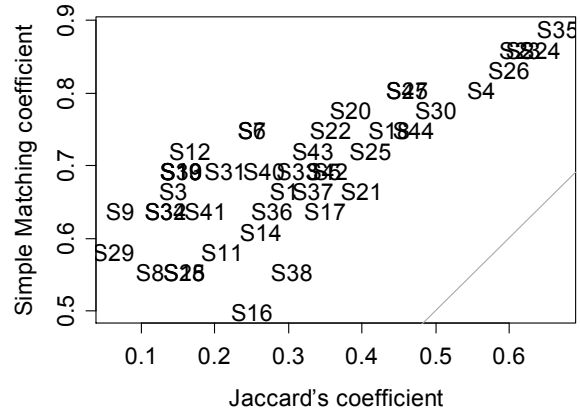
Next, let's just compare the similarities between site S1 and all others (recall that the dataset contains 45 sites (objects)) and add site labels instead of points, by typing:

```
plot(sp.jac[1:45],sp.sim[1:45],xlab="Jaccard's coefficient",ylab="Simple
Matching coefficient",type="n")

text(sp.jac[1:45],sp.sim[1:45],row.names(speocc))

abline(0,1,col="darkgray")
```

This should produce a plot resembling the one presented on below. How might we explain the large discrepancy for site S29?

Repeat the above comparisons for site similarities based on Sørenson's coefficient and other coefficients of interest. Have fun exploring and explaining these differences. Use this information and your knowledge of the statistical properties of the coefficients to select an appropriate similarity index for examining patterns of community composition based on species presence/absence.





CALCULATING COEFFICIENTS OF (DIS)SIMILARITY FOR CONTINUOUS <u>SPECIES</u> DATA (SPECIES ABUNDANCE, DENSITY)

Let's calculate the (dis)similarity among sites according to their fish community composition based on species abundance. From the **`vegan`** library we can use the function **`vegdist`** which can calculate a number of similarity/dissimilarity coefficients commonly used in ecology.

Its usage is:

```
vegdist(x, method="bray", binary=FALSE, diag=FALSE, upper=FALSE,
      na.rm = FALSE, ...)
```

The relevant arguments are:

1. x – the data matrix to be analyzed
2. method - Dissimilarity index, partial match to "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "altGower ", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao" or "mahalanobis".
3. binary - Perform presence/absence standardization before analysis using decostand. True or false (default)

For additional documentation type "?vegdist" at the R prompt.

Let's try calculating a dissimilarity (distance) matrix based on the Bray-Curtis coefficient:

```
sp.bray <- vegdist(speabu, method="bray")
```

Type **`sp.bray`**. The first upper left cells in the Bray-Curtis distance matrix should be:

```
      S1        S2        S3
S2    0.8418879
S3    0.8635236 0.7457213
S4    0.9900867 0.9854015 0.8963415
```

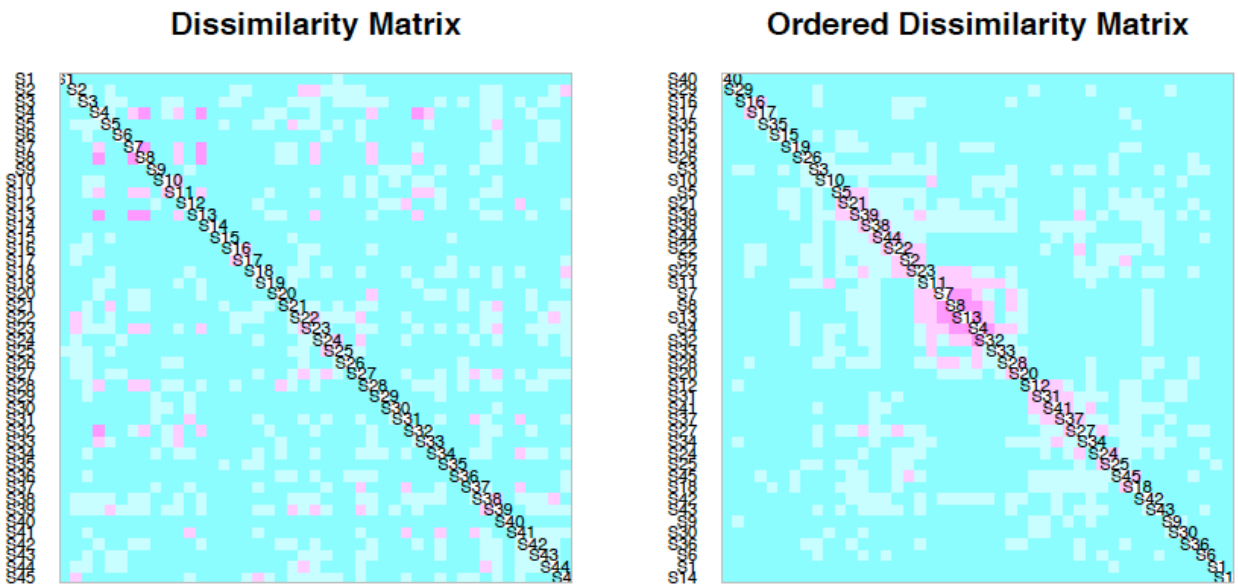
Recall that these are distance values, therefore a value of 1 indicates two sites that have zero similarity in species abundance.

When you have many objects in your dataset it may be more useful to display them in a way that emphasizes their main features.

Borcard et al. (2011) provides a nice function called "coldiss.R" that color codes distances in a heat map with (or without) re-ordering the records by dissimilarity. The function calls upon other functions in the gclus library. So, remember to source the coldiss.R file and install the gclus library!

```
coldiss(sp.bray,nc=4,byrank=FALSE,diag=TRUE)
```

The resulting plot (below) displays the raw and ordered dissimilarity matrices, where magenta are dissimilarities close to 0 and cyan are dissimilaries close to 1. The number of colors is set using “nc”, byrank=T refers to equal-sized categories and byrank=F refers to equal-length categories. If diag=T then object labels are displayed on the diagonal.



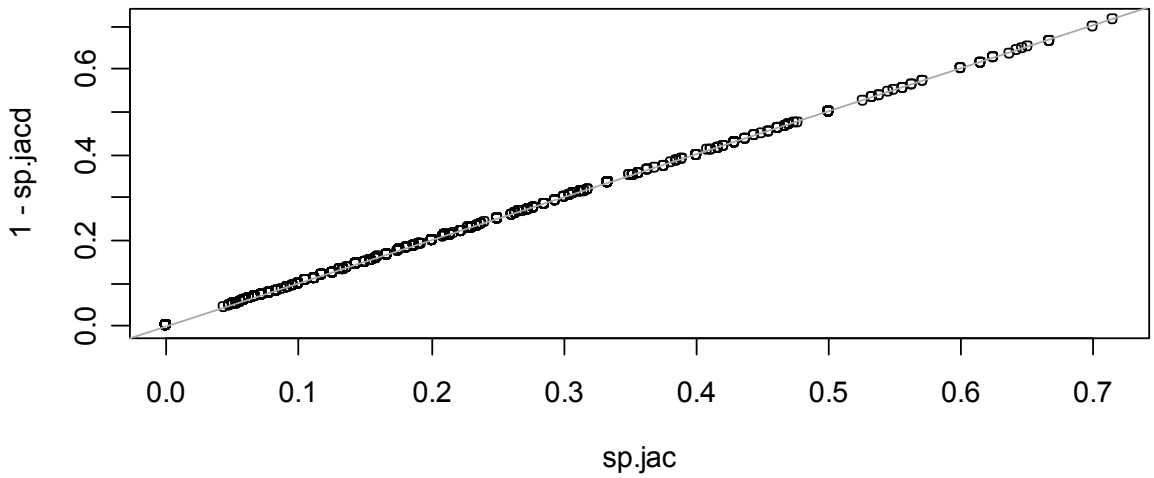
Also note that you can use the **vegdist** function to calculate many of the similarity coefficients that we previously explored. For example, you can calculate a distance matrix based on Jaccard’s coefficient, by typing:

```
sp.jacd<-vegdist(speocc, method="jaccard")
```

You should find that **sp.jacd** = 1- **sp.jac** because the only difference is that the first matrix expresses the values as distances and the second matrix contains similarity values. Quickly plot the results to confirm this, by typing:

```
plot(sp.jac,1-sp.jacd)
abline(0,1,col="darkgray")
```

This will return the following in the graphic window:



There are a number of alternative dissimilarity and distance metrics that are appropriate for asymmetrical species data (abundance) that are worth exploring. For example, we can use **Chord distance** or **Hellinger distance** on the raw species abundance datasets by first transforming the dataset and then calculating the Euclidean distance matrix. This is discussed in detail in the lecture.

For Chord's distance:

```
speabu.norm<-decostand(speabu, method='nor')
sp.chordd<-dist(speabu.norm)
```

For Hellinger's distance:

```
speabu.hel<-decostand(speabu, method='hel')
sp.held<-dist(speabu.hel)
```

CALCULATING COEFFICIENTS OF SIMILARITY FOR MIXED DATA TYPES

In many instances you will be interested in assessing the similarity among objects that are described by a suite of mixed descriptor types, including continuous, nominal and ordinal. When faced with this challenge, I suggest that you seriously consider Gower's similarity coefficient (see lecture notes).

Using the fish species trait data, let's explore the utility of Gower's similarity coefficient. First, let's take a look at the trait data by typing:

```
str(spetrait)
```

This will confirm that the data matrix contains both continuous and categorical data. Next, let's calculate Gower's similarity using the **`daisy`** function in the **`cluster`** library. NOTE: You will not be able to use the **`sim`** or **`vegdist`** functions because the trait matrix contains non-numeric data. Also, the Gower calculations in these Packages do not handle multiclass variables and do not handle missing values correctly (at least this was the case the last time I checked). The **`daisy`** function computes all the pairwise dissimilarities (distances) between observations in the data set.

Its usage is:

```
daisy(x, metric = c("euclidean", "manhattan", "gower"),
stand = FALSE, type = list())
```

The relevant arguments are:

4. x – the data matrix to be analyzed
5. metric - character string specifying the metric to be used. The currently available options are "euclidean" (the default), "manhattan" and "gower".

For additional documentation type “?daisy” at the R prompt.

Let’s try calculating a dissimilarity (distance) matrix based on Gower’s coefficient:

```
sptr.gower <- daisy(spetrait,metric="gower")
```

Type **sptr.gower**. The first upper left cells in the Gower distance matrix should be:

```
          BANDDART   BANDSCUL   BLACDACE   BLUEGILL
BANDSCUL 0.17464214
BLACDACE 0.28047433 0.28189602
BLUEGILL 0.52564155 0.61766608 0.62016722
BLUECHUB 0.32943933 0.32146386 0.10316266 0.67120222
```

We can quickly produce a heat map to look at similarities/diferences between species according to their traits. Here is just the ordered heat map.

```
coldiss(sptr.gower,nc=4,byrank=FALSE,diag=TRUE)
```

Makes sense. Taxonomically similar species have low distances (or high trait similarities) as indicated by the magenta color.

Special note: **gowdis()** function in the **FD** library is the most complete function to compute Gower's coefficient. It computes the distance for mixed variables, including sasymmetrical binary variables. Variable weights can be specified! Check it out.

CALCULATING COEFFICIENTS OF (DIS)SIMILARITY FOR CONTINUOUS DATA

For continuous data in which double-zeros are meaningful (i.e., environment data, species measurements or traits, etc …) we can calculate dissimilarities between objects based on any number of distance coefficients. Let's calculate environmental dissimilarity among the MAHA sites according to Euclidean distance (the most commonly used coefficient). Start by typing:

```
env.euc <- vegdist(envdata, method="euclidean")
```

Type **env.euc.** The first upper left cells in the Euclidean distance matrix should be:

```
      S1          S2          S3
S2    905.84678
S3    197.51321   720.34388
S4    193.87537   1086.18669  386.26245
```

To calculate pairwise Mahalanobis distances between sites is slightly more involved. Lots of solutions on the Internet. I suggest you check the following if you are interested in this distance metric.

http://stats.stackexchange.com/questions/65705/pairwise-mahalanobis-distance

---

CONVERTING SIMILARITY TO DISSIMILARITY (DISTANCE) OR VICE VERSA

Remember that when dealing with matrices, it is possible to change a similarity matrix (S) into a dissimilarity matrix (D) by applying the following transformations:

$$D = 1 - S \qquad D = \sqrt{(1 - S)} \qquad D = \sqrt{(1 - S^2)}$$

Note that we did this previous using Jaccard's coefficient.

CONVERTING CORRELATION TO DISTANCE

At times you might want to convert a correlation matrix to a distance matrix in order to perform subsequent multivariate analysis. Of course, you cannot use the transformations listed above because correlation values will vary between -1 and 1, thus resulting in negative distances when R<0. Not possible! Therefore I recommend the following transformation:

$$D = \sqrt{(2 - 2*\text{correlation value})}$$

Let's do this using the environmental data by typing:

```
env.dis <- sqrt(2-2*cor(envdata))
```

OPTIONAL READINGS

Baroni-Urbani, C., and M.W. Buser. 1976. Similarity of binary data. Syst. Zool. 25:251-259.

Bloom, S.A. 1981. Similarity indices in community studies: potential pitfalls. Marine Ecology – Progress Series 5: 125-128.

Bray, J.R., and J.T. Curtis. 1957. An ordination of the upland forest communities of southern Wisconsin. Ecological Monographs 27: 325-349.

Chao, A., R.L. Chazdon, R.K. Colwell, and T-J Shen. 2005. A new statistical approach for assessing similarity of species composition with incidence and abundance data. Ecology Letters 8: 148-159.

Faith, D.P., P.R. Minchin and L. Belbin. 1987. Compositional dissimilarity as a robust measure of ecological distance. Vegetatio 69: 57-68.

Jackson, D.A., K.M. Somers, and H.H. Harvey. 1989. Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence? American Naturalist 133:436-453.

Pavoine, S., Vallet, J., Dufour, A.B., Gachet, S., and Daniel, H. 2009. On the challenge of treating various types of variables: application for improving the measurement of functional diversity. Oikos 118: 391-402.

Wolda, H. 1981. Similarity indices, sample size, and diversity. Oecologia 50: 296-302.

EXERCISE

Purpose
In this exercise, we will explore a large group of association coefficients based on the notion of similarity and dissimilarity.

Tasks

1. Compute the similarity among the objects of your data using an appropriate suite of association coefficients
    a. What is your ecological interpretation of associations among objects according to the descriptors (e.g., among stream sites according to their fish communities)
    b. How do the similarity matrices differ? Can you interpret these differences based on what you know about the equations underpinning the coefficients?
2. Compute the (dis)similarity among the descriptors of your data using an appropriate suite of association coefficients
    a. What is your ecological interpretation of associations among descriptors?
    b. How do the similarity matrices differ? Can you interpret these differences based on what you know about the equations underpinning the coefficients?
3. Compare association matrices for a single coefficient using raw and transformed data.
    a. How do your results differ? Think about simple graphically and numerical ways to present these comparisons.
4. Summarize patterns of similarity in your own dataset using the approaches described in Chapter 2.
    a. Can you use your results to identify potential outliers or influential variables?

Properties of distance coefficients calculated from a variety of similarity coefficients. Reprinted from Table 7.2. and Table 7.3 of Legendre and Legendre (2012).

Table 7.2 Continued.

Similarity coefficient	$D = 1 - S$ metric, etc.	$D = 1 - S$ Euclidean	$D = \sqrt{1 - S}$ metric	$D = \sqrt{1 - S}$ Euclidean
$S_{13} = \frac{1}{2}\left[\frac{a}{a+b} + \frac{a}{a+c}\right]$ (eq. 7.16)	semimetric	No	No	No
$S_{14} = \frac{a}{\sqrt{(a+b)\,(a+c)}}$ (Ochiai; eq. 7.17)	semimetric	No	Yes	Yes
$S_{15} = \sum w_j s_j / \sum w_j$ (Gower; eq. 7.21)	metric	No	Yes	Likely* (S_1)
$S_{16} = \sum w_j s_j / \sum w_j$ (Estabrook & Rogers; eq. 7.22)	metric	No	Yes	Likely* (S_1)
$S_{17} = \frac{2W}{A+B}$ (Steinhaus; eq. 7.24)	semimetric	No	Likely* (S_8)	Likely* (S_8)
$S_{18} = \frac{1}{2}\left[\frac{W}{A} + \frac{W}{B}\right]$ (Kulczynski; eq. 7.25)	semimetric	No	No* (S_{13})	No* (S_{13})
$S_{19} = \sum w_j s_j / \sum w_j$ (Gower; eq. 7.26)	metric	No	Yes	Likely
$S_{20} = \sum w_j s_j / \sum w_j$ (Legendre & Chodorowski; 7.27)	metric	No	Yes	Likely* (S_7)
$S_{21} = 1 - \chi^2$ *metric* (eq. 7.28)	metric	Yes	Yes	Yes
$S_{22} = 2\,(\sum d)\,/\,n\,(n-1)$ (Goodall; eq. 7.29)	semimetric	No	–	–
$S_{23} = 1 - p(\chi^2)$ (Goodall; eq. 7.30)	semimetric	No	–	–
$S_{26} = (a + d/2)\,/\,p$ (Faith, 1983; eq. 7.18)	metric	No	Yes	Yes

* These results follow from the properties of the corresponding binary coefficients (coefficient numbers given), when continuous variables are replaced by binary variables.
– Property unknown for this coefficient.

Table 7.2 Some properties of distance coefficients calculated from the similarity coefficients presented in Section 7.3. These properties (from Gower & Legendre, 1986), which will be used in Section 9.3, strictly apply when there are no missing data.

Similarity coefficient	$D = 1 - S$ metric, etc.	$D = 1 - S$ Euclidean	$D = \sqrt{1 - S}$ metric	$D = \sqrt{1 - S}$ Euclidean
$S_1 = \frac{a+d}{a+b+c+d}$ (simple matching; eq. 7.1)	metric	No	Yes	Yes
$S_2 = \frac{a+d}{a+2b+2c+d}$ (Rogers & Tanimoto; eq. 7.2)	metric	No	Yes	Yes
$S_3 = \frac{2a+2d}{2a+b+c+2d}$ (eq. 7.3)	semimetric	No	Yes	No
$S_4 = \frac{a+d}{b+c}$ (eq. 7.4)	nonmetric	No	No	No
$S_5 = \frac{1}{4}\left[\frac{a}{a+b} + \frac{a}{a+c} + \frac{d}{b+d} + \frac{d}{c+d}\right]$ (eq. 7.5)	semimetric	No	No	No
$S_6 = \frac{a}{\sqrt{(a+b)\,(a+c)}} \frac{d}{\sqrt{(b+d)\,(c+d)}}$ (eq. 7.6)	semimetric	No	Yes	Yes
$S_7 = \frac{a}{a+b+c}$ (Jaccard; eq. 7.10)	metric	No	Yes	Yes
$S_8 = \frac{2a}{2a+b+c}$ (Sørensen; eq. 7.11)	semimetric	No	Yes	Yes
$S_9 = \frac{3a}{3a+b+c}$ (eq. 7.12)	semimetric	No	No	No
$S_{10} = \frac{a}{a+2b+2c}$ (eq. 7.13)	metric	No	Yes	Yes
$S_{11} = \frac{a}{a+b+c+d}$ (Russell & Rao; eq. 7.14)	metric	No	Yes	Yes
$S_{12} = \frac{a}{b+c}$ (Kulczynski; eq. 7.15)	nonmetric	No	No	No

Table 7.3 Some properties of the distance coefficients described in Section 7.4.

Distance coefficient	D metric, etc.	D Euclidean	$\sqrt{D}$ metric	$\sqrt{D}$ Euclidean
D_1 (Euclidean distance; eq. 7.32)	metric	Yes	Yes	Yes
D_2 (average distance; eq. 7.34)	metric	Yes	Yes	Yes
D_3 (chord distance; eqs. 7.35, 7.36)	metric	Yes	Yes	Yes
D_4 (geodesic metric; eq. 7.37)	metric	No	Yes	Yes
D_5 (Mahalanobis generalized distance; eq. 7.38)	metric	Yes	Yes	Yes
D_6 (Minkowski metric; eq. 7.43)	metric	*	–	–
D_7 (Manhattan metric; eq. 7.44)	metric	No	Yes	Yes
D_8 (mean character difference; eq. 7.45)	metric	No	Yes	Yes
D_9 (index of association; eqs. 7.47, 7.48)	metric	No	Yes	Yes
D_{10} (Canberra metric; eq. 7.49)	metric	No	Yes	Yes
D_{11} (coefficient of divergence; eq. 7.51)	metric	Yes	Yes	Yes
D_{12} (coefficient of racial likeness; eq. 7.52)	nonmetric	No	No	No
D_{13} (nonmetric coefficient; eq. 7.57)	semimetric	No	Yes	Yes
D_{14} (percentage difference; eq. 7.58)	semimetric	No	Yes	Yes
D_{15} (χ^2 metric; eq. 7.54)	metric	Yes	Yes	Yes
D_{16} (χ^2 distance; eq. 7.55)	metric	Yes	Yes	Yes
D_{17} (Hellinger distance; eq. 7.56	metric	Yes	Yes	Yes
D_{18} (distance between species profiles; eq. 7.53)	metric	Yes	Yes	Yes
D_{19} (modified mean character difference; eq. 7.46)	semimetric	No	No	No

* The result depends on the exponent r.
– Not tested for all exponents r.

4. Hierarchical Cluster Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Perform hierarchical agglomerative clustering, plot dendrograms

R Packages: vegan, cluster, pvclust, NbClust, clusteval
R Source: biostats

BACKGROUND

Cluster analysis encompasses several multivariate techniques that are used to group objects into categories based on their (dis)similarities. The aim is both to minimize within-group variation and maximize between-group variation in order to reveal well-defined categories of objects, and therefore reduce the dimensionality of the data set to a few groups of rows (James and McCulloch 1990, Legendre and Legendre 1998). Therefore, this approach is generally recommended when distinct discontinuities instead of continuous differences (i.e. gradients) are expected between samples (objects) because cluster analysis mostly aims at representing partitions in a data set.

Cluster analysis of a data matrix proceeds in two steps. First, a relevant association coefficient must be chosen to measure the association (similarity or dissimilarity) among objects or among variables. Second, the calculated association matrix is represented as a horizontal dendrogram or tree (hierarchical clustering) or as distinct groups of objects (k-means clustering), based on specific rules to aggregate objects. For ecologists, the power of cluster analysis derives from the existence of different types of (dis)similarity coefficients. The choice of appropriate and ecologically meaningful association coefficients is particularly important because it directly affects the values that are subsequently used for the categorization of objects.

Two general approaches to clustering are commonly used: hierarchical clustering and non-hierarchical partitioning. In hierarchical clustering, a linkage rule to form clusters and the numbers of clusters that best suit the data have to be determined *a priori* or subjectively based on the clustering results. Clusters, which are nested rather than mutually exclusive here, are formed by either agglomerative or divisive algorithms. The basic theory of agglomerative hierarchical analysis is to partition a similarity/distance matrix of n samples (or variables) into a hierarchy of k ($k < n$) groups using a clustering strategy. It begins with each object or variable, assigns these objects or variables to a cluster, and agglomerates them in a hierarchy of larger and larger clusters until finally a single cluster contains all the objects or variables. The five commonly used clustering techniques are single-linkage, complete-linkage, centroid clustering, average-linkage, and the minimum-variance clustering. Divisive cluster analysis begins with all objects together in a single cluster and successively divides the objects into a hierarchy of smaller clusters, either using a single variable (i.e. monothetic divisive cluster analysis) or many variables (i.e. polythetic divisive cluster analysis). These two strategies do not necessarily yield the same clusters.

SET-UP

In this exercise you will be working with the MAHA environment dataset and species abundance dataset.

Let's first set-up your R work session by defining the current work directory to your folder of choice and loading the vegan, cluster and pvclust Packages. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Second, import the MAHA environment dataset and MAHA species abundance dataset, by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

CLUSTER ANALYSIS IN R

The cluster package in R includes a wide spectrum of methods, corresponding to those presented in Kaufman and Rousseeuw (1990). Curiously, the methods all have the names of women that are derived from the names of the methods themselves. Of the partitioning methods, **pam** is based on partitioning around mediods, **clara** is for clustering large applications, and **fanny** uses fuzzy analysis clustering. Of the hierarchical methods, **agnes** uses agglomerative nesting, **diana** is based on divisive analysis, and **mona** is based on monothetic analysis of binary variables. Other functions include **daisy**, which calculates dissimilarity matrices, including Gower's coefficient. Basic agglomerative hierarchical clustering can be performed with **hclust** in the Stats library and the **agnes** function in the Cluster library. The **hclust** function will be the main focus of this chapter.

CONDUCT HIERARCHICAL AGGLOMERATIVE CLUSTERING

Hierarchical clustering is typically used when the objective is to establish groups of similar objects and simultaneously describe the relationships among the groups. The results are typically depicted in a dendrogram, which is a tree-like plot depicting the agglomerative or divisive sequence of fusions or splits, respectively. Each step in cluster analysis is described below.

1. Compute an appropriate dissimilarity/distance matrix

A necessary pre-cursor to any cluster analysis is to compute an appropriate dissimilarity/distance matrix (although sometimes this step is included in the clustering algorithm and therefore does not need to be done separately). For example, let's calculate the Euclidean distance based on site environmental characteristics (see R documentation: Ecological Resemblance for the entire list of association coefficients). First, let's standardize the environment matrix (mean = 0, variance = 1; i.e., z-score) in order to control for the fact that variables are measured on different scales (recall the nasty properties of Euclidean distance), by typing:

```
envdata.tra<-data.stand(envdata,method='standardize',margin='column',plot=F)
```

How might you quickly double-check that the function actually transformed your data appropriately? Hint: type **mean(envdata.tra[,1])**. Now, let's calculate the Euclidean distance matrix, by typing:

```
site.eucd<-vegdist(envdata.tra,method='euclidean')
```

2. Compute hierarchical clustering

Now we are ready to build the hierarchy using one of two basic strategies: agglomerative and divisive; and many alternatives within each of these general approaches. Here, we'll focus on the clustering function from the stats package (this package is automatically loaded when R is opened). The function performs hierarchical agglomerative clustering on dissimilarity matrices supplied by the user.

Its usage is:

```
hclust(d, method = "complete", members=NULL)
```

The arguments are:

- d – the dissimilarity matrix to be analyzed
- method – the agglomeration method to be used. Choices are:

    - ward.D2 – Implements Ward's (1963) minimum variance method, compared to ward.D which does not (see Murtagh and Legendre 2014).
    - single – Single linkage.
    - complete – Complete linkage. The default method.
    - average
    - mcquitty
    - median
    - centroid
- members – vector assigning sample units to clusters. Optional; used to construct a dendrogram beginning in the middle of the cluster analysis (i.e., beginning with the clusters already specified by this argument).

Like always, you can type "**?hclust**" at the **R** prompt for additional documentation.

Try typing any one of the following commands:

```
sitecl.ave<-hclust(site.eucd,method='average')
sitecl.cen<-hclust(site.eucd,method='centroid')
sitecl.med<-hclust(site.eucd,method='median')
sitecl.sin<-hclust(site.eucd,method='single')
sitecl.com<-hclust(site.eucd,method='complete')
sitecl.ward<-hclust(site.eucd,method='ward.D2')
```

3. Examining clustering results

The results of the clustering process are given in the corresponding object. For the object produced by hclust(), we can print a simple summary of the agglomeration sequence using the hclus.table() function in biostats, by typing:

```
hclus.table(sitecl.ave)
```

The table gives the sequence of fusions and the corresponding distance at which the fusion took place. In the agglomeration sequence, if a number j in the row is negative, then the single observation |j| is merged at this stage. If j is positive, then the merger is with the cluster formed at stage j of the algorithm. The Euclidean distance between objects/clusters is also provided. Here is some of the output:

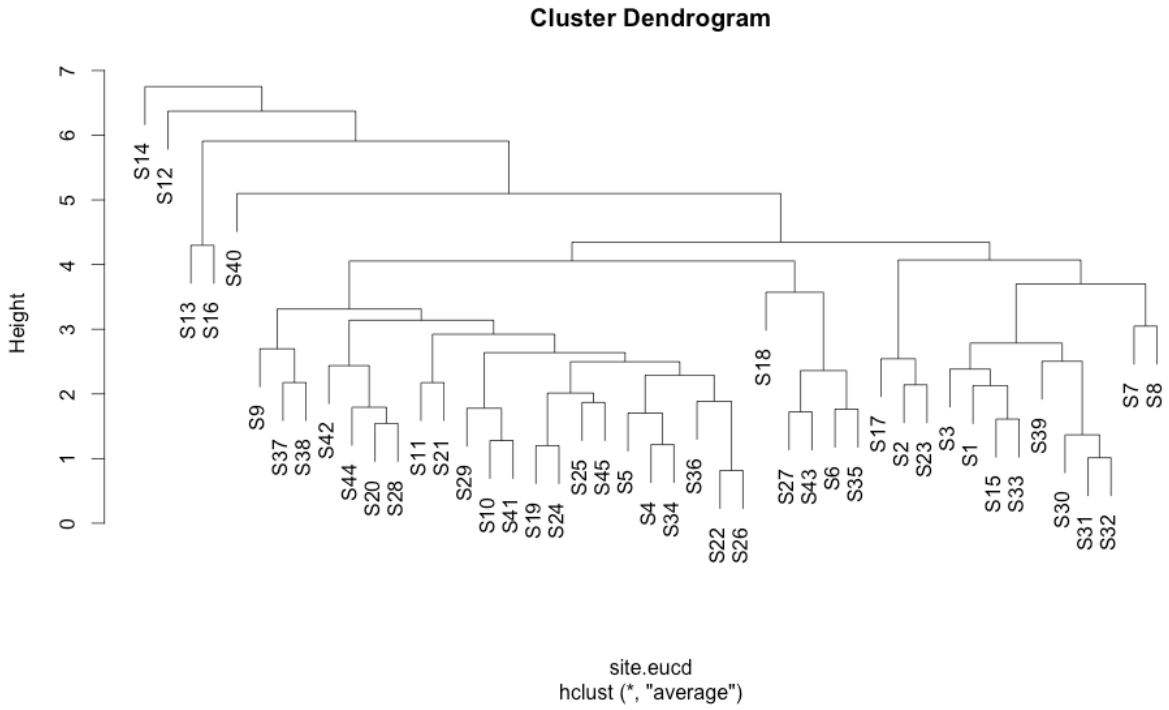
```
$dist.method
[1] "euclidean"

$method
[1] "average"

$cluster.table
   no. clusters entity entity  distance
1            44    -22    -26 0.8159428
2            43    -31    -32 1.0138844
3            42    -19    -24 1.2000233
4            41     -4    -34 1.2166423
5            40    -10    -41 1.2790520
…
```

Dendrogram: The primary way to inspect the results from hierarchical clustering is using a dendrogram plot. The dendrogram depicts the fusion or splitting sequence used to create the cluster hierarchy and graphically portrays the relationships both within and among clusters by showing the distance at which

objects fuse together or split apart. To create a dendrogram from the average-linkage agglomerative clustering produced by the hclust() function, type:

```
plot(sitecl.ave)
```

Your dendrogram should look identical to the one rendered below.



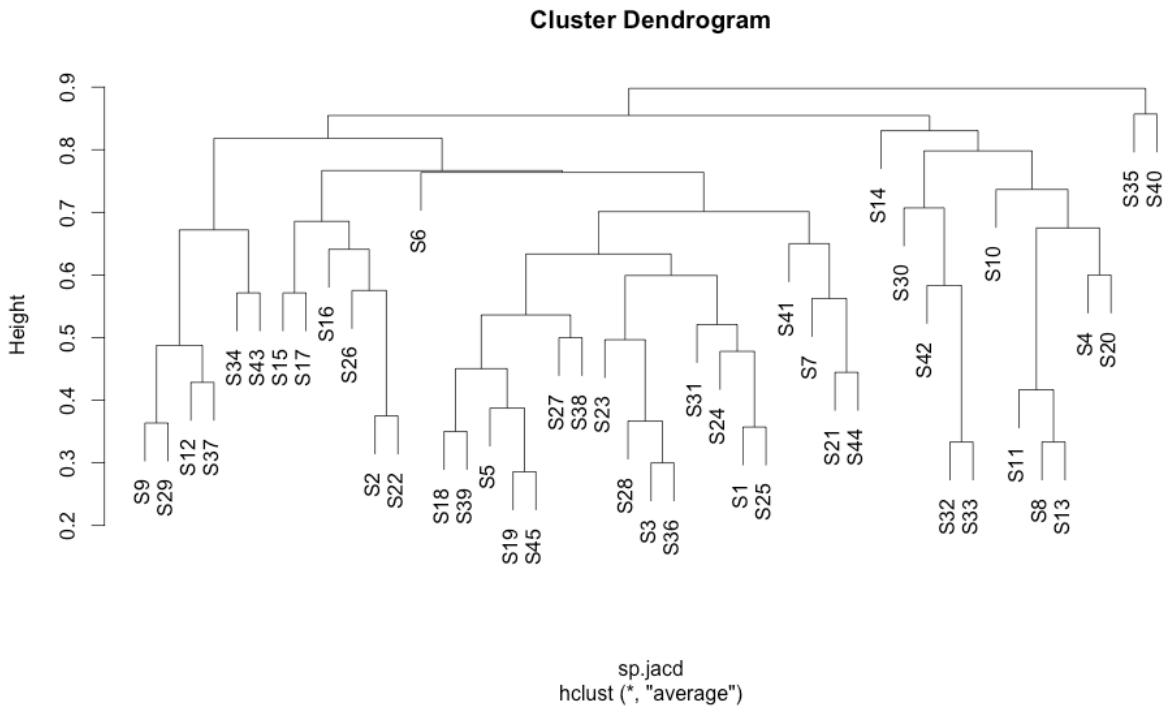
You can make your dendrogram fancier by typing:

```
plot(sitecl.ave,main='Average-linkage
Dendrogram',xlab='Sites',ylab='Euclidean distance',hang=-1)
```

Quickly, let's cluster the sites according to fish species presence/absence based on Jaccard's similarity coefficient! Type the following:

```
speocc <- data.trans(speabu,method='power',exp=0,plot=F)
sp.jacd<-vegdist(speocc,method="jaccard")
specl.ave<-hclust(sp.jacd,method='average')
plot(specl.ave)
```

You should be able to follow each of the steps above. You will obtain the following for the average-linkage agglomerative clustering.



Okay, let's focus again on the site-environment analysis. Based on the dendrogram, we may be ready to decide on the number clusters to retain for our final solution. We can show on the dendrogram the final cluster membership using the rect.hclust() function, either on the basis of the height of the dendrogram (i.e., dissimilarity value) or a specified number of clusters, as follows:

But first, let's re-plot the dendrogram of environmental distances based on average linkage.

```
plot(sitecl.ave)
rect.hclust(sitecl.ave,k=9)
```

This identifies 9 clusters in the dendrogram.

```
plot(sitecl.ave)
rect.hclust(sitecl.ave,h=6)
```

This identifies clusters in the dendrogram according to Euclidean distance = 6. That is, all objects must be within a Euclidean distance of 6 to be considered members of the same cluster.

Based on this analysis we can use the cutree() function to "cut" the dendrogram into clusters, by typing:

```
sitecl.class<-cutree(sitecl.ave,k=9)
```

This cuts the dendrogram into 9 clusters.

Note, the cutree() creates a vector called **`sitecl.class`** that contains the cluster membership (i.e., cluster ID) for each observation in the data set, which we will use below. Neat!

4. Cluster Diagnostics: Evaluating the cluster solution

Agglomerative coefficient: The agglomerative coefficient measures the amount of clustering structure of the dataset (see Lecture Notes). If observations quickly agglomerate into distinct clusters that later agglomerate into a single cluster at much greater dissimilarities, the coefficient will approach 1. By contrast, datasets with weak or no clustering will have an agglomerative coefficient approaching zero.

The agglomerative coefficient can be computed by typing:

```
coef.hclust(sitecl.ave)
```

This should return a value of 0.668. You can substitute any of the suitable cluster objects that you computed above.

Cophenetic correlation coefficient: One way to evaluate how well the cluster hierarchy represents the original object-by-object dissimilarity space is to compute the cophenetic correlation coefficient. The cophenetic distance between two observations that have been clustered is defined to be the intergroup dissimilarity at which the two observations are first combined into a single cluster (this is produced by the cophenetic function used below). The cophenetic correlation is the correlation between the dissimilarities in the original p-dimensional space and the cophenetic distances. It can be argued that a dendrogram is an appropriate summary of some data if the correlation between the original distances and the cophenetic distances is high. Otherwise, it should simply be viewed as the description of the output of the clustering algorithm.

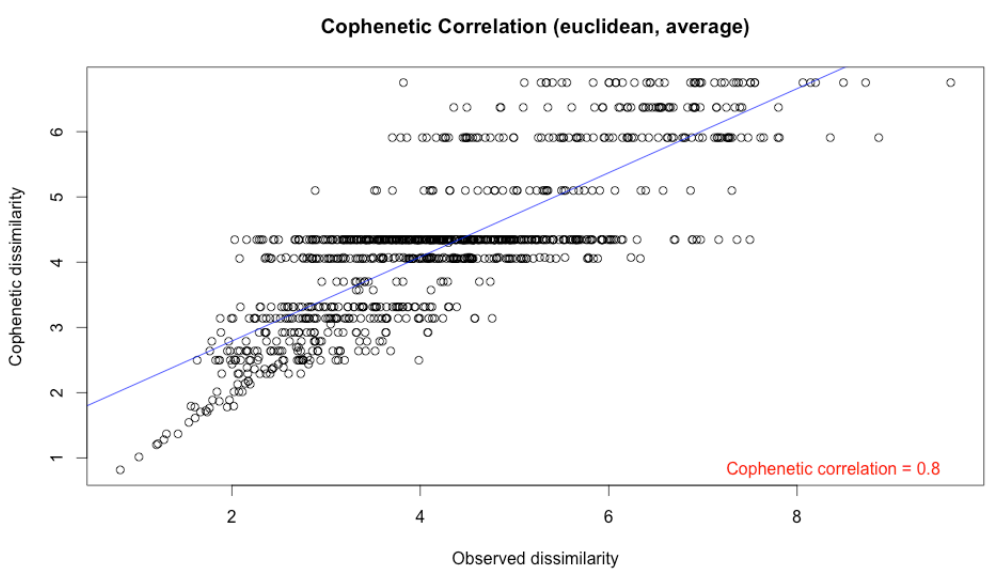
The cophenetic correlation can be computed directly by typing:

```
cor(site.eucd,cophenetic(sitecl.ave))
```

This should return a value of 0.803. Alternatively, we can plot the cophenetic relationship using the hclus.cophenetic() function in biostats by typing:

```
hclus.cophenetic(site.eucd, sitecl.ave)
```

This will also return a value or 0.80 and produce the following:

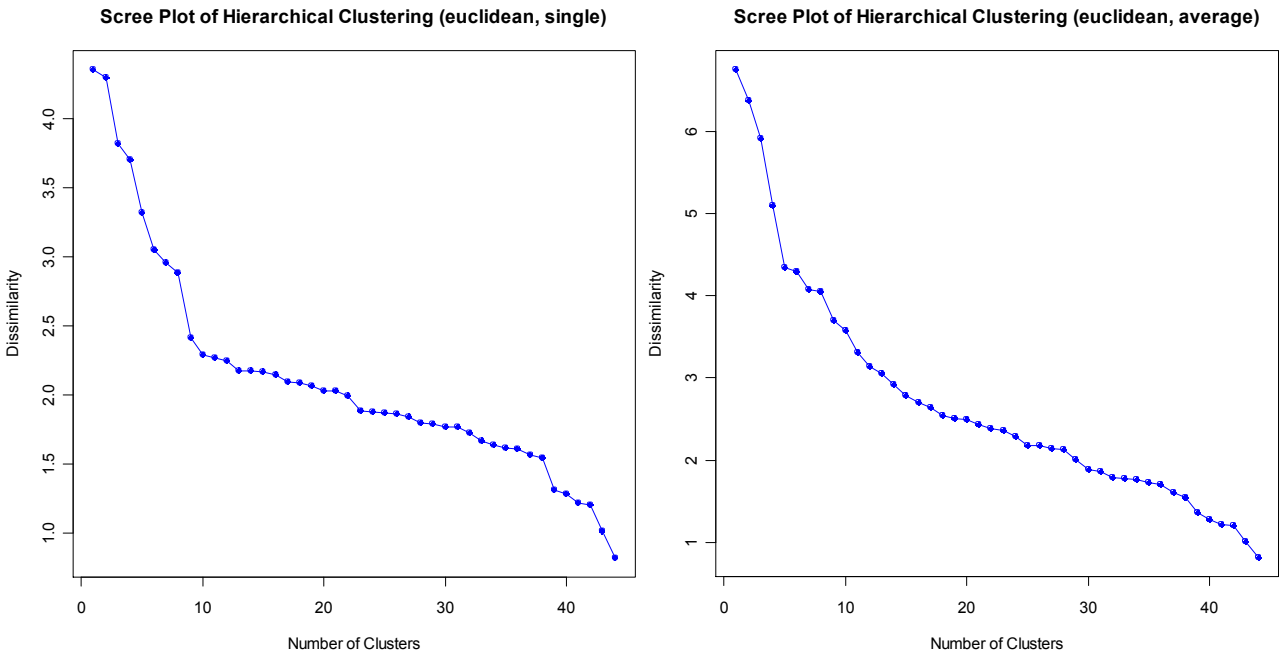


5. Cluster Diagnostics: Determining the number and stability of clusters

Scree plot: An important way to summarize the hierarchical structure of the clusters and determine the number of clusters is using a scree plot. A scree plot depicts the dissimilarity value of the fusion/split against the number of clusters. For an agglomerative hierarchical clustering, as the number of clusters increases, the dissimilarity value at which clusters fuse together typically decreases monotonically. In this case, the scree curve is read from right to left. Abrupt changes in the scree slope or a pronounced "elbow" in the scree plot is an indication of where there is a large change in dissimilarity as clusters fuse together. This is a logical level at which to "cut" the dendrogram. Let's look at the scree plot for clustering based on single linkage and average linkage, by typing:

```
hclus.scree(sitecl.sin)
hclus.scree(sitecl.ave)
```

You should obtain the following:



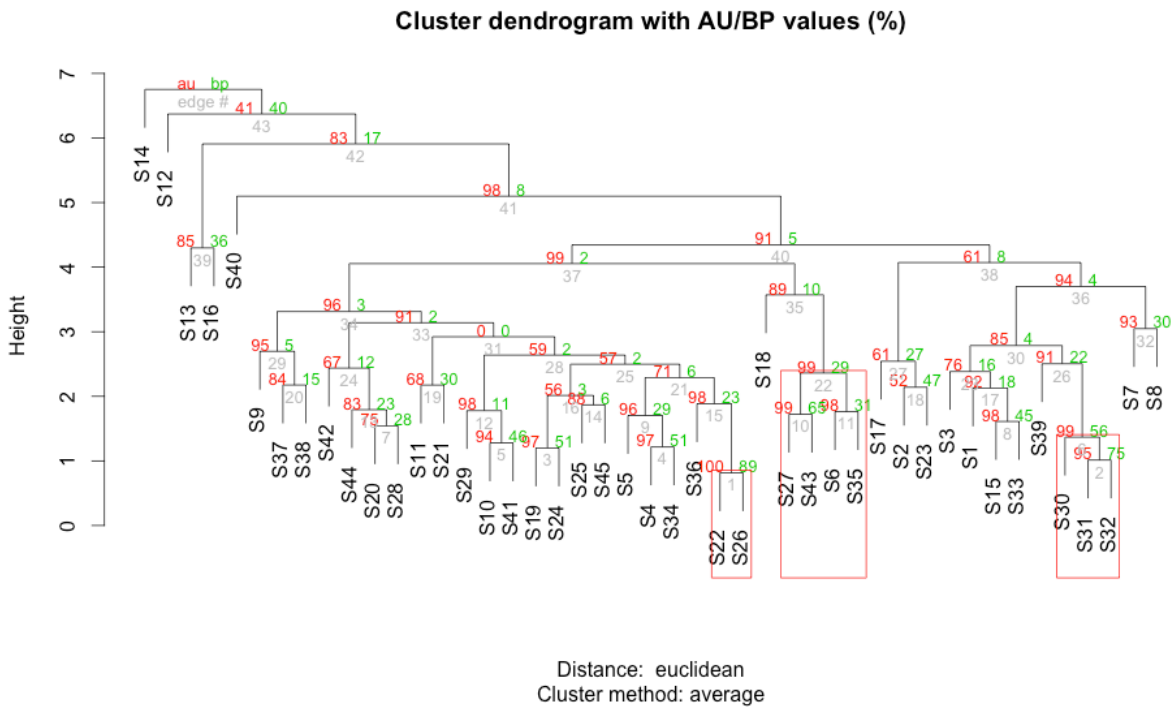
Bootstrap Test: It is always recommended that you evaluate the stability of the derived clusters. By doing so one can ask: Are the clusters cohesive? Do they remain intact with minor perturbations to the data? For example, if we select a subset of objects at random, do the original clusters remain intact, albeit reduced in size depending on which observations made it into the random subset? If we create a bootstrap sample from the original data set by sampling observations at random with replacement, do the original clusters remain intact in the bootstrap samples? Many such functions have been developed for DNA data analysis. Here, we will use the pvclust() function in the pvclust library to address these questions. This function calculates p-values for hierarchical clustering via multiscale bootstrap resampling. Begin by typing:

```
clus.stab<-pvclust(t(envdata.tra),method.hclust="average",method.dist
="euclidean",nboot=50)
```

Take note that we perform this computation on the tranpose of the environmental data matrix because this function assumes that objects are columns NOT rows. We have selected average-linkage clustering based on euclidean distance, and only run 50 bootstrap samples to save computational time. Next we can view the plot and highlight significant clusters at a critical P <0.01, by typing:

```
plot(clus.stab)
pvrect(clus.stab,alpha=0.99)
```

Based on the 50 random bootstrap samples the following dendrogram was produced with statistically significant clusters in red boxes (gray in your handout). The values at each node represent two types of p-values: au (Approximately Unbiased) p-value on the left and bp (Bootstrap Probability) value on the right. The AU p-value, which is computed by multiscale bootstrap resampling, is a better approximation to an unbiased p-value than the BP value, which is computed by normal bootstrap resampling. Note that the AU and BP values, as well as the statistical significance boxes, that you produce will be different that the values in the dendrogram below because your analysis will be based on a different set of bootstrapped samples. Also note that this method might not work for smaller sample sizes – just be warned!



Index-based Assessment: A number of indices are available to select the "optimal" number of clusters in a database. NbClust package provides 30 indices for determining the number of clusters. The indices are based largely on Milligan and Cooper (1985), with the addition of the Dunn index (Dunn, 1974), Silhouette statistic (Rousseeuw, 1987), Gap statistic (Tibshirani, 2001), Dindex (Lebart, 2000), SD index and SDbw index (Halkidi et al., 2000, 2001), and Statistic of Hubert (Hubert and Arabie, 1985).

Its usage is:

NbClust(data, diss = "NULL", distance = "euclidean", min.nc = 2, max.nc =15, method = "ward", index = "all", alphaBeale = 0.1)

The arguments are:

- data - matrix or dataset (the only mandatory argument)

- diss - dissimilarity matrix to be used. By default, diss="NULL", but if it is replaced by a dissimilarity matrix, distance should be "NULL".
- Distance - the distance measure to be used to compute the dissimilarity matrix. This must be one of: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski" or "NULL". By default, distance="euclidean". If the distance is "NULL", the dissimilarity matrix (diss) should be given by the user. If distance is not "NULL", the dissimilarity matrix should be "NULL".
- min.nc - minimal number of clusters, between 1 and (number of objects - 1)
- max.nc - maximal number of clusters, between 2 and (number of objects - 1), greater or equal to min.nc. By default, max.nc=15.
- method - the cluster analysis method to be used. This should be one of: "ward", "single", "complete", "average", "mcquitty", "median", "centroid", "kmeans".
- Index - the index to be calculated. This should be one of : "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sdindex", "dindex", "sdbw", "all" (all indices except GAP, Gamma, Gplus and Tau), "alllong" (all indices with GAP, Gamma, Gplus and Tau included).
- alphaBeale - significance value for Beale's index.

Like always, you can type "**`?NbClust`**" at the **R** prompt for additional documentation.

Let's begin by typing:

```
NbClust(envdata.tra,distance="euclidean",min.nc=2, max.nc=10,
method="ward.D2")
```

Results returned include the amount of support for different number of clusters and the number of clusters based on the majority rule. In this case, the best number of clusters is 8.

[All.index] contains the values of indices for each partition of the dataset obtained with a number of clusters between min.nc and max.nc.

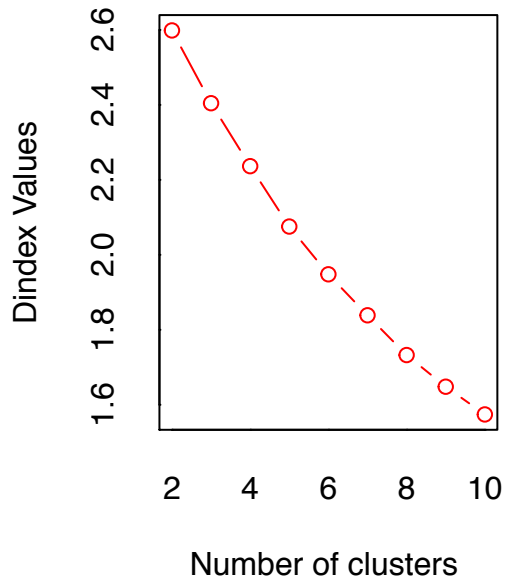
[All.CriticalValues] Critical values of some indices for each partition obtained with a number of clusters between min.nc and max.nc.

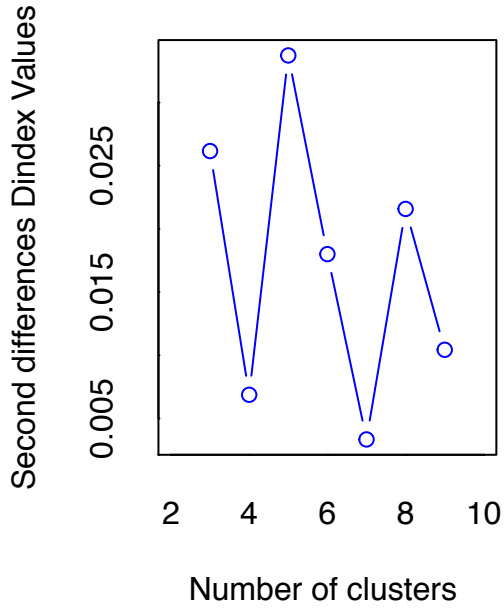
[Best.nc] Best number of clusters proposed by each index and the corresponding index value.

```
$Best.nc
                  KL          CH          Hartigan    CCC
Number_clusters   8.0000      2.0000      8.0000      10.0000
Value_Index       1.7601      10.9103     1.8579      1.653
…
```

You can use the majority rule to select the number of clusters or consider only the indices that have performed best in simulations studies (e.g., Milligan and Cooper 1985). Top-5 indices in Milligan and Cooper (1985) are: CH index, Duda index, Cindex, Gamma and Beale index. For binary data, Dimitriadou et al. (2002) found that Ratkowsky –Lance indiex performed the best.

Finally, the package alo creates plots according to the Hubert index and D Index.





In case you are thirsty for more, other functions exist to compute clustering indices! The function **`clustIndex()`** in the cclust library is a nice option.

6. Cluster Diagnostics: Comparing two clusters

There are times that you will be interested in comparing different cluster solutions that are based on different cluster algorithms (e.g., complete vs. average linkage) or datasets (e.g., species vs. environmental clustering) according to the same set of objects.

Many approaches exist, but here we will ultilize the clusteval library.

Let's use the cluster_similarity() function.

Its usage is:

```
cluster_similarity(labels1, labels2, similarity = c("jaccard", "rand"), method = "independence")
```

The arguments are:

- labels1 – a vector of n clustered labels
- labels2 – a vector of n clustered labels
- similarity – the similarity statistics to calculate
- method – the model under which the statistic was derived

Details:

To calculate the similarity, the 2x2 contingency table is computed, consisting of the following cells:

n_11 - the number of observation pairs where both observations are comembers in both clusterings

n_10 - the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 - the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 - the number of observation pairs where neither pair are comembers in either clustering

Let's begin by determining object membership according to three different clustering algorithms by typing:

```
sitecl.class.ave<-cutree(sitecl.ave,k=9)
sitecl.class.sin<-cutree(sitecl.sin,k=9)
sitecl.class.com<-cutree(sitecl.com,k=9)
```

Next, let's calculate the commonly used Rand Index (Rand 1971) to estimate the similarity between the cluster-derived memberships based on average linkage and single linkage.

```
cluster_similarity(sitecl.class.ave, sitecl.class.sin, similarity = "rand",
method = "independence")
```

This will return a value of 0.577 (range from 0 to 1), indicating moderate similarity.

Next, let's calculate the Rand Index to estimate the similarity between the cluster-derived memberships based on complete linkage and single linkage.

```
cluster_similarity(sitecl.class.com, sitecl.class.sin, similarity = "rand",
method = "independence")
```

This will return a value of 0.461, indicating lower similarity. Given what you know about the clustering algorithms, does this result make sense?

We can also look at the co-membership table for this last comparison (see n_11, n_10, n_01, n_00 above) by typing:

```
comembership_table(sitecl.class.sin,sitecl.class.com)
```

7. Cluster Diagnostics: Describing the clusters

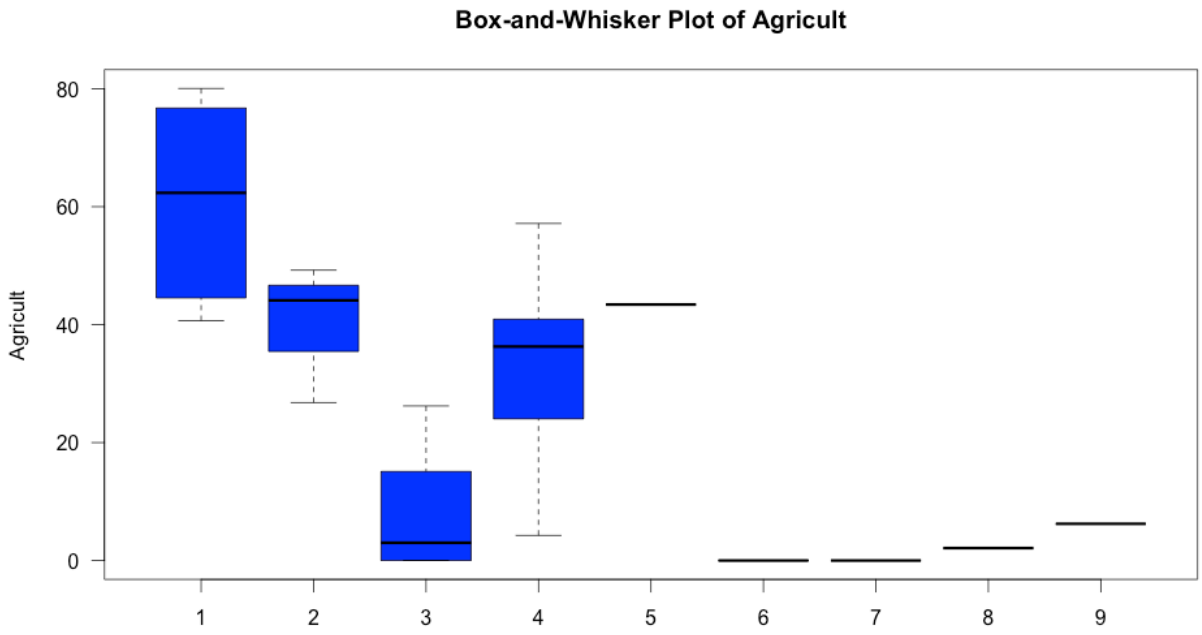
Once we have examined the clusters, we may want to describe the differences among clusters in ecological terms. In particular, we may wish to explore univariate differences among clusters based on the intrinsic variables used to cluster the samples or on extrinsic variables not used in the clustering. Begin by typing:

```
envdata.new<-cbind(sitecl.class,envdata)
```

This will bind the cluster classes (computed above) with the original dataset. You will notice that cluster membership in the 9 classes is now the first column of envdata.new matrix. Now we use box.plots()to produce separate box-and-whisker plots for each variable for each cluster, by typing:

```
box.plots(envdata.new,by='sitecl.class')
```

Here is the plot that is returned for agricultural landuse. What can you infer regarding differences between clusters?



OTHER PACKAGES FOR HIERARCHICAL AGGLOMERATIVE CLUSTERING

The **`agnes()`** function (an abbreviation for agglomerative nesting) in the Cluster library provides an alternative to **`hclus()`**. The only difference is that **`agnes()`** provides a novel graphical display and allows for conducting flexible beta clustering. Note that **`agnes()`** does not have an option for the median or centroid linkages. Feel free to explore the **`agnes()`** output by typing (don't forget to plot the results!).

Flexible Beta Clustering

In recent years many ecologists have expressed a preference for a hierarchical clustering algorithm call "flexible Beta" developed by Australian ecologists Lance and Williams (1966). Lance and Williams determined that because all hierarchical agglomerative algorithms operate similarly, but differ in the calculation of multi-member dissimilarity, that it was possible to generalize the algorithms to a single algorithm with four parameters, alpha_1, alpha_2, beta, and gamma. In Table 8.8 of Legendre and Legendre (1998) the parameter values to recreate many of the classical algorithms are presented. This table is reproduced below:

Values of parameters α_h, α_i, β, and γ in Lance and Williams' general model for combinatorial agglomerative clustering. Modified from Sneath & Sokal (1973) and Jain & Dubes (1988).

Clustering method	α_h	α_i	β	γ	Effect on space A
Single linkage	1/2	1/2	0	−1/2	Contracting*
Complete linkage	1/2	1/2	0	1/2	Dilating*
UPGMA	$\frac{n_h}{n_h+n_i}$	$\frac{n_i}{n_h+n_i}$	0	0	Conserving*
WPGMA	1/2	1/2	0	0	Conserving
UPGMC	$\frac{n_h}{n_h+n_i}$	$\frac{n_i}{n_h+n_i}$	$\frac{-n_h n_i}{(n_h+n_i)^2}$	0	Conserving
WPGMC	1/2	1/2	−1/4	0	Conserving
Ward's	$\frac{n_h+n_g}{n_h+n_i+n_g}$	$\frac{n_i+n_g}{n_h+n_i+n_g}$	$\frac{-n_g}{n_h+n_i+n_g}$	0	Conserving
Flexible	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	$-1 \le \beta < 1$	0	Contracting if $\beta \approx 1$ Conserving if $\beta \approx -.25$ Dilating if $\beta \approx -1$

* Terms used by Sneath & Sokal (1973).

Of more interest here is a special case where alpha_1 = alpha_2 = 0.625, beta = -0.25, and gamma = 0. Observe that not only are alpha_1 and alpha_2 equal, it is also the case that beta = 1 - alpha_1 - alpha_2. This set of parameters returns a satisfactory result. To perform a flexible-beta using agnes(), the call is a little more complicated than we have seen for other clustering algorithms. We need to specify alpha_1, alpha_2, beta, and gamma in the call, using the par.method argument.

```
sitecl2.flb <- agnes(site.eucd, method='flexible',par.method=c(0.625,0.625,-0.25))
```

The defaults, however, are that alpha_1 = alpha_2, beta = 1 - alpha_1 - alpha_2, and gamma = 0. So we can simply specify alpha_1 and get the desired results.

The plot function for an object of class "agnes" has two panels. First, the "banner plot" is given, where the height of fusions is shown as a white bar on a red background. The second plot gives the more traditional dendrogram. Both plots include the agglomerative coefficient.

You can return the class memberships using the same approach implemented above. Type:

```
sitecl.class<-cutree(sitecl2.flb,k=9)
```

CONDUCT HIERARCHICAL DIVISIVE CLUSTERING

In contrast to agglomerative hierarchical clustering, there are only a limited number of divisive hierarchical algorithms that are available in R. The diana() algorithm constructs a hierarchy of clusterings, starting with one large cluster containing all n objects. Clusters are divided until each cluster contains only a single object. At each stage, the cluster with the largest diameter is selected

(the diameter of a cluster is the largest dissimilarity between any two of its objects). To divide the selected cluster, the algorithm first looks for its most disparate object (i.e., which has the largest average dissimilarity to the other objects of the selected cluster). This object initiates the "splinter group". In subsequent steps, the algorithm reassigns objects that are closer to the "splinter group" than to the "old group". The result is a division of the selected cluster into two new clusters. To compute a divisive hierarchical clustering using diana(), type the following:

```
sitecl3.dia<-diana(site.eucd)
```

CONDUCT MONOTHETIC HIERARCHICAL DIVISIVE CLUSTERING

If you are interested in conducting monothetic clustering (i.e., using a single descriptor as the basis for the partitioning of the data at each step), then I suggest you check out the mona() function in the Cluster library. This function returns a list representing a divisive hierarchical clustering of a dataset with binary variables only. Try typing the following:

```
sitecl4.mona<-mona(speocc)
```

OPTIONAL READINGS

Belbin, L., and C. McDonald. 1993. Comparing three classification strategies for use in ecology. Journal of Vegetation Science 4:341-348.

Gauch, H.G. and R.H. Whittaker. 1981. Hierarchical classification of community data. Journal of Ecology 69: 537-557.

Hunter, J.C. and R.A. McCoy. 2004. Applying randomization tests to cluster analysis

Jackson, D.A., Somers, K.M. and H.H. Harvey. 1992. Null models and fish communities: evidence of non-random patterns. Am. Nat. 139: 930-951.

Kaufman, L.A. and P.J. Rousseeuw. 1990. Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, New York, NY, US.

Milligan, G. W., and M. C. Cooper. 1985. An examination of procedures for determining the number of clusters in a data set. Psychometrika 50:159–179.

Strauss, R.E. 1982. Statistical significance of species clusters in association analysis. Ecology 63: 634-639.

EXERCISE

Purpose

In this exercise, we will explore the advantages and disadvantages of different clustering algorithms, and engage in the necessary steps of cluster analysis, including (1) computing an appropriate dissimilarity/distance matrix, (2) computing hierarchical clustering, (3) examining clustering results, (4) evaluating the cluster solution, (5) determining the number and stability of clusters, and (6) describing the clusters.

Tasks

1. Perform a cluster analysis on your own data (or the class data), using centroid, average, single, and complete linkage. Produce a dendrogram for each clustering method.
    a. Which clustering method performed best?
    b. For the best clustering method, how many clusters would you define?

2. Using a selected clustering outcome (from above), perform a statistical test on each variable testing the null hypothesis of equal cluster means.
3. Data standardization/transformation can play an important role in cluster analysis. Examine this yourself by comparing cluster results from raw vs. transformed data.
4. Flexible Beta clustering provides ultimate flexibility in terms of controlling the outcome of cluster analysis. Compare the dendrograms produced by using different beta values (holding alpha and gamma constant) with respect to space contracting, conserving and dilating.

You will often be required to write a series of commands to help explore the power of multivariate statistics. Below is how you might answer question #4 above.

```
# Create a vector with values from -1 to 1 by increments of 0.01
beta<-seq(-1,1,by=0.01)
size=length(beta)

# Create an empty vector to store the agglomerative coefficient values (note
# that we are defining the size equal to beta)
aggcoef<-vector(length=size)

# Create a loop to cluster the data using increasing values of beta and
# record the agglomerate coefficient for each cluster solution
for (j in 1:size) {
temp<-agnes(site.eucd, method='flexible',par.method=c((1- beta[j])/2,(1-
beta[j])/2,beta[j]))
aggcoef[j]<-temp$ac
}

plot(beta,aggcoef,xlab="Beta coef.",ylab="Agglomerate coef.")
```

5. Non-hierarchical Cluster Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Perform non-hierarchical divisive clustering

R Packages: vegan, cluster, pvclust
R Source: biostats

BACKGROUND

Two general approaches to clustering are commonly used: hierarchical clustering (discussed in the previous chapter) and non-hierarchical partitioning. Non-hierarchical cluster analysis merely assigns each object or variable to a cluster in a multidimensional space without showing the interrelationships between objects. K-means is the most commonly used non-hierarchical clustering approach used in ecology. In k-means clustering, objects are assigned to k clusters (k being defined in advance), based on their nearest Euclidean distance to the mean of the clusters. The mean of the cluster is iteratively recalculated until no more assignments are made and cluster means fall below a predefined cut-off value or until the iteration limit is reached. Different means for each cluster are ideally obtained for each dimension used in the analysis, as indicated by high F-values from the respective analyses of variance. Unlike hierarchical clustering, k-means clustering does not require prior computation of dissimilarity matrix among objects and is therefore more adapted to large data sets (e.g. few thousand objects) where computing power is an issue. However, the method is quite sensitive to outliers, which are usually removed before performing the analyses (Legendre and Legendre 1998).

SET-UP

In this exercise you will be working with the MAHA environment dataset and species abundance dataset.

Let's first set-up your R work session by defining the current work directory to your folder of choice and loading the vegan and cluster Packages. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Second, import the MAHA environment dataset and MAHA species abundance dataset, by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

CONDUCT NON-HIERARCHICAL DIVISIVE CLUSTERING

Non-hierarchical divisive clustering is typically used when the objective is to summarize a very large data set into a much smaller set of groups, where the sole criterion for clustering samples is maximizing within-cluster similarity. Often, this is a precursor to subsequent analyses based on the newly formed clusters, and it may even involve non-hierarchical clustering to elucidate relationships among the newly formed clusters.

K-MEANS CLUSTER ANALYSIS IN R

Although most cluster functions in R operate on a dissimilarity/distance matrix, K-means is based directly on the object-by-descriptor dataset. Recall from your lecture that K-means clustering is based on the Euclidean distance among objects. However, we still must account for the situations where the descriptors (variables) are measured on different scales. In this example, we need to standardize the environment matrix (mean = 0, variance = 1) in order to control for this fact, by typing:

```
envdata.tra<-data.stand(envdata,method='standardize',margin='column',plot=F)
```

Now, let's calculate the Euclidean distance matrix, by typing:

```
site.eucd<-vegdist(envdata.tra,method='euclidean')
```

Determine the number of clusters

In contrast to hierarchical clustering, with non-hierarchical clustering we typically need to specify the number of clusters sought. In some cases we will know the number of clusters we want to identify. However, the majority of the time we don't have a good idea of how many clusters to expect and we would like characteristics of the data to determine how many clusters to keep. In this case, we can construct a scree plot of some objective criterion against the number of clusters and look for indications of the "best" number of clusters to retain. One possible criterion is the average within-cluster dissimilarity, which declines monotonically as the number of clusters increases; it typically declines rapidly at first and then levels off somewhat. An abrupt leveling off or "elbow" in the scree plot provides an indication of the number of clusters to keep.

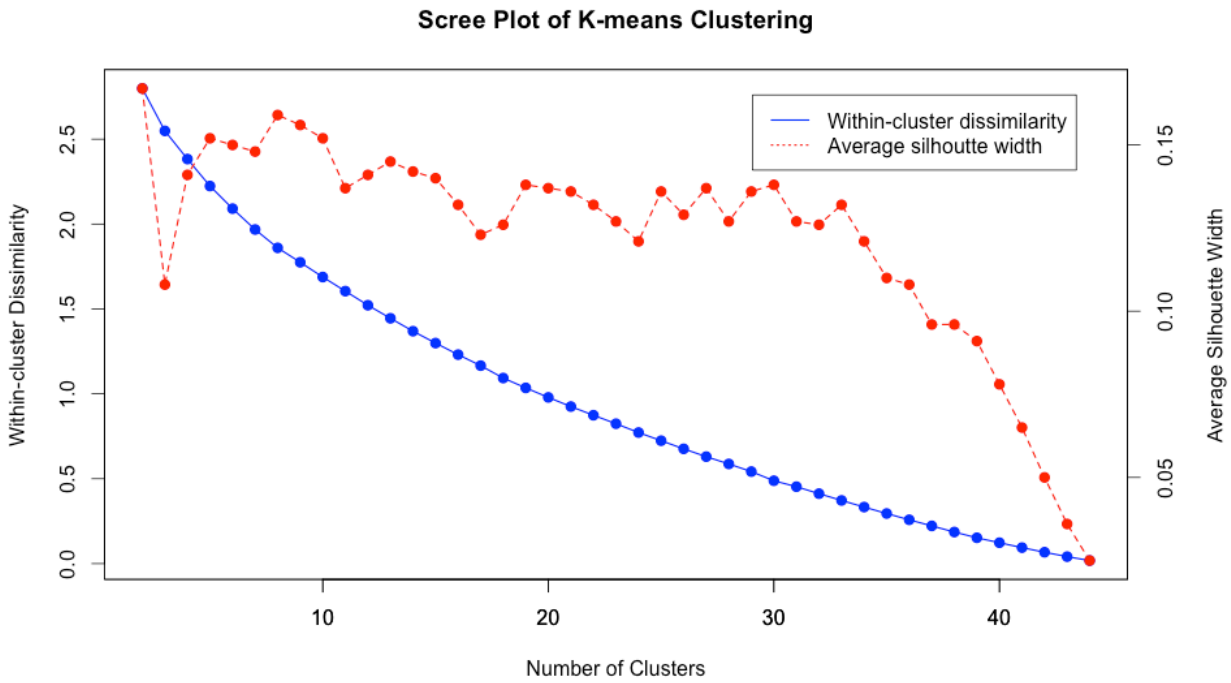
Another criterion is the average silhouette width. For each observation i, the silhouette width s(i) is defined as follows:

> Put a(i) = average dissimilarity between i and all other points of the cluster to which i belongs (if i is the only observation in its cluster, s(i) = 0 without further calculations). For all other clusters C, put d(i,C) = average dissimilarity of i to all observations of C. The smallest of these d(i,C) is b(i) = min_C d(i,C), and can be seen as the dissimilarity between i and its "neighbor" cluster, i.e., the nearest one to which it does not belong.
>
> Finally, s(i) = [b(i) - a(i)] / max[a(i), b(i)].

Observations with a large s(i) (almost 1) are very well clustered, a small s(i) (around 0) means that the observation lies between two clusters, and observations with a negative s(i) are probably placed in the wrong cluster. We can plot the average silhouette width for all observations against the number of clusters in a scree plot. We expect the average silhouette width to be highest when the number of clusters equals the natural clustering of the data. We can produce a scree plot of both objective criteria using the **`nhclus.scree()`** function in biostats, as follows:

```
nhclus.scree(site.eucd,max.k=44)
```

Note: the scree plot will depict the objective criteria for 2 to max.k number of clusters. Obviously, max.k cannot exceed the number of sample observations N and it probably makes no sense for max.k to approach N. The following plot should be produced:



From the scree plot you will notice that the average silhouette width is highest for 2 and 8 cluster solutions. You might also notice that there doesn't appear to be a clear elbow in the dissimilarity curve, therefore this provides little guidance for selecting the number of clusters in the data.

Compute K-means clustering

The K-means algorithm is a non-hierarchical method to cluster objects based on attributes into K partitions. In other words, given *n* objects in a *p*-dimensional space, this algorithm determines a partition of objects into K groups, or clusters, such that the objects within each cluster are more similar to each other than to objects in other clusters. The function **`kmeans`** (Stats library) is available to perform this cluster analysis. Several algorithms have been developed that accomplish the overall objective of identifying stable clusters and **`R`** gives you the option of specifying which algorithm to use. Nuances in each algorithm may result in slightly different cluster assignment patterns but, in general, the default algorithm setting "Hartigan-Wong" provides the most stable solutions.

The function's usage is:

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm)

The arguments are:

- **x** – object-by-descriptor matrix
- **centers** – the number of clusters desired.
- **inter.max** – maximum number of iterations the clustering algorithm should be employed (default is 10).
- **nstart** – number of random starts the algorithm should be initiated with to ensure that a stable clustering solution is reached (default is 1).
- **algorithm** – specify which clustering algorithm to use. Options include
    - Hartigan-Wong (default if algorithm is not specified)
    - Lloyd
    - Forgy
    - MacQueen

For additional documentation type "**?kmeans**" at the **R** prompt.

For this demonstration we'll use the transformed environmental dataset (see above). Now, let's perform k-means clustering. We'll request 8 clusters (based on the maximum average silhouette value from above – not including a cluster number of 2) and use a reasonably large number of iterations and random starts to ensure that a stable solution is converged upon. Begin by typing:

```
sitecl.kmeans<-kmeans(envdata.tra,centers=8,iter.max=10000,nstart=25)
```

Let's look at the results. Type:

```
sitecl.kmeans
```

Your results should look something like what appears below (why does it look different?)

```
K-means clustering with 8 clusters of sizes 5, 2, 6, 8, 7, 15, 1, 1

Cluster means:
    Sinuosity       Slope     WDRatio    SubEmbed     HabQual       Elev
1  0.34211414 -0.39076272  0.70826713 -0.70880097  1.26784743  0.4639215
2  0.78144578 -0.13148937 -0.22203311 -0.81992249  0.03486025 -1.4233593
3 -0.45547873  0.04137107  1.37458021  0.02519612 -0.16430947  0.9483609
…

      RoadDen   Agricult   HumanUse   BasinAre
1 -0.10433884  0.3062505  0.1452294  1.9760121
2  3.96706642 -0.9392717  1.3381329 -0.2896444
3 -0.46556915 -0.6329201 -0.6848619  0.7085082
…

Clustering vector:
 S1  S2  S3  S4  S5  S6  S7  S8  S9 S10 S11 S12 S13 S14 S15 S16 S17 S18
  5   5   4   6   6   1   4   4   3   6   6   7   2   8   4   2   5   1
S19 S20 S21 S22 S23 S24 S25 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36
  6   6   6   5   5   6   6   5   1   6   6   4   4   4   4   6   1   5
S37 S38 S39 S40 S41 S42 S43 S44 S45
  3   3   3   3   6   3   1   6   6

Within cluster sum of squares by cluster:
[1] 15.96343  9.23792 25.08302 29.99266 20.21755 46.68402  0.00000
[8]  0.00000

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

From the above output, "Cluster means" indicates the average value of the descriptor within each cluster; "Clustering vector" identifies the cluster identity of each site.

You can extract specific results from the newly created object **`sitecl.kmeans`** by referencing the available components (this is true of any object in R). For example, you can obtain the site classifications by typing:

```
sitecl.kmeans$cluster
```

Remember that you can also create a new table containing the extracted information by setting a new object name equal to the previous comment. For example, type:

```
sitecl.class<-sitecl.kmeans$cluster
```

Finally, also remember that you can examine the available components of any R object by typing:

```
names(sitecl.kmeans)
```

The analysis above yields a single partition with a pre-defined number of groups. You can try several solutions with different k values by using the function **`cascadeKM()`** which is a wrapper for **`kmeans()`**.

The function's usage is:

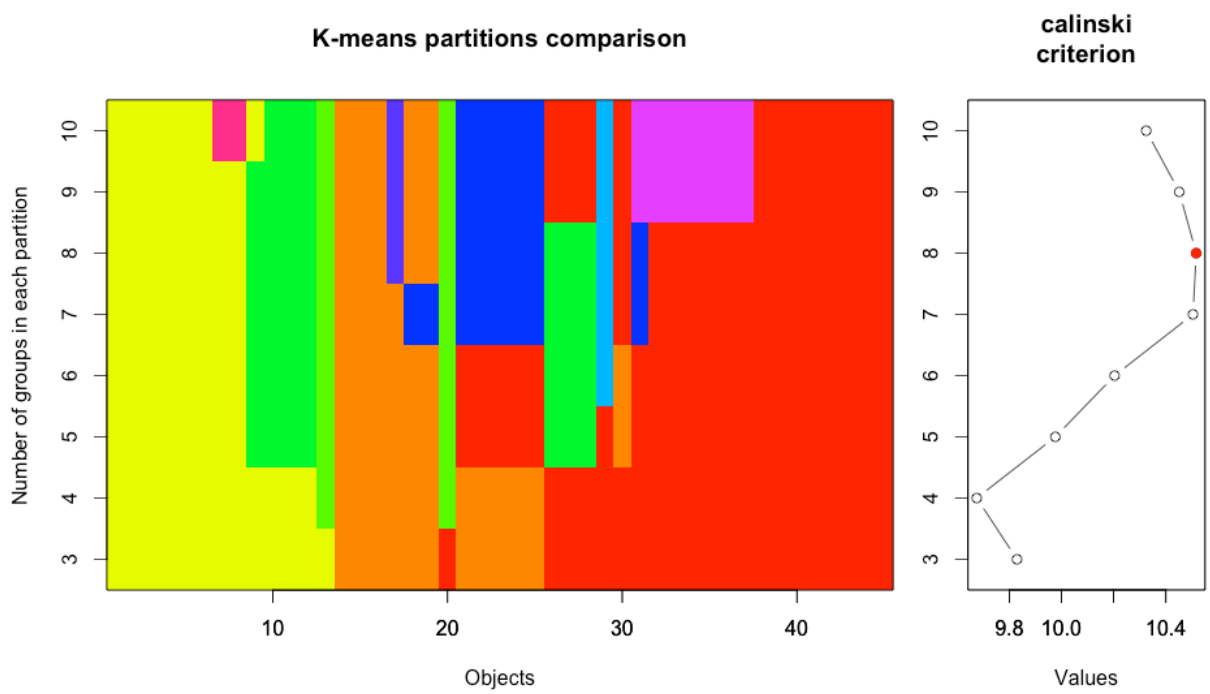
cascadeKM(data, inf.gr, sup.gr, iter = 100, criterion = "calinski")

The arguments are:

- Data - The data matrix. The objects (samples) are the rows.
- inf.gr - The number of groups for the partition with the smallest number of groups of the cascade (min).
- sup.gr - The number of groups for the partition with the largest number of groups of the cascade (max).
- iter - The number of random starting configurations for each value of K.
- criterion - The criterion that will be used to select the best partition. The default value is "calinski", which refers to the Calinski-Harabasz (1974) criterion. The simple structure index ("ssi") is also available. Other indices are available in function clustIndex (package cclust). In our experience, the two indices that work best and are most likely to return their maximum value at or near the optimal number of clusters are "calinski" and "ssi".

Let's begin by typing:

```
sitecl.kmeans.cas<-cascadeKM(envdata.tra,inf.gr=3,sup.gr=10,iter=100)
plot(sitecl.kmeans.cas,sortg=T)
```



The plot shows the group attributed to each object for each partition (rows of the graph). The groups are represented by different colors; there are two colors for k=2, three colors for k=3, etc … The graph to the right illustrates the values of the Calinski criterion for different values of the k. You can type:

```
sitecl.kmeans.cas$results
```

if you want the values that are contained in this plot. The Calinski index represents the F-statistic comparing the among-group to the within-group sum of squares of the partition. In other words, larger values provide more support for a particular number of clusters/groups. How many clusters does this analysis support?

Evaluating the number of clusters using randomization

Recall from Chapter 4 that it is possible to evaluate the "goodness-of-fit" of the selected number of clusters using a randomization test. To do so, one must typically write a short script (i.e., series of commands) in R. The idea of writing a script can be taunting, so let's take this one step at a time.

The randomization test will involve the following steps:

1. Define initial parameters
2. Perform K-means clustering on the original data and calculate the within-groups sum-of-squared error (SSE) for the number of cluster solutions selected by the user
3. For a number of different cluster numbers, randomize the order of the columns in the original matrix to create a random dataset
4. Perform K-means clustering on the random data and calculate the within-groups SSE
5. Repeat steps 3 and 4, and large number of times (here, I selected 999)
6. Compare the actual SSE to the distribution of random SEE

```
no.clus<-10
no.ran<-999
no.var<-10
```

Maximum number of clusters to consider
Number of randomizations to perform
Number of descriptors in the dataset

```
sse<-matrix(,nrow=no.ran+1,ncol=no.clus-1)
randata<-matrix(,nrow=nrow(envdata.tra),ncol=ncol(envdata.tra))
```

Creates an empty matrix to store within SSE values from the k-means on the original data (first row) and randomized data (rows 2, 3 … no.ran), where each column is a different number of clusters

Also creates an empty matrix to store the randomized dataset

```
for (j in 3:no.clus) {
```

For loop that repeats all commands between { and } a total of j times, where j = the number of clusters to consider starting with 3

```
sitecl.kmeans<-kmeans(envdata.tra,centers=j,iter.max=1000,nstart=5)
sse[1,j-1]<-sum(sitecl.kmeans$withinss)
```

Performing k-means using the original data, calculating the SSE and storing results in the first row and the j-1 column (e.g., column 1 for a two cluster solution)

```
for (k in 1:no.ran) {
```

For loop that repeats all commands between { and } a total of k times, where k = the number of randomizations

```
      for (c in 1:no.var) {
            randata[,c]<-envdata.tra[,c][sample(nrow(envdata.tra))]
      }
```

For loop that randomly selects values from each column (i.e., randomization)

```
      ran.kmeans<-kmeans(randata,centers=j,iter.max=1000,nstart=5)
      sse[k+1,j-1]<-sum(ran.kmeans$withinss)
```

Performing k-means using the random data, calculating the SSE and storing results in the k+1 row and j-1 column

```
      }
}
```

Closing the cluster (j) and randomization (k) for loops

```
diff<-colMeans(sse)-sse[1,]
plot(seq(2,no.clus,by=1),diff,xlab="No. clusters",ylab="mean(ranSSE)-
actualSSE")
```

Calculating the difference between the average SSE based on the random k-means and the true SSE based on the actual data. Then the results are plotted.

Based on this information, what level of support do we have for the effectiveness of k-means based on different numbers of clusters? How many clusters would you pick for subsequent interpretation?

PARTITIONING AROUND MEDOIDS (PAM)

A number of other non-hierarchical divisive clustering algorithms exist, although these approaches are not commonly used in ecology. For example, the **`pam()`** or **`clara()`** functions in the Cluster library are methods for partitioning (clustering) of the data into k clusters 'around medoids', which is a more robust version of K-means clustering (Kaufman and Rousseeuw 1990). A particularly nice property is that PAM (Partitioning around medoids) allows clustering with respect to any specified distance metric (not just Euclidean distance, which k-means is based on). In addition, the medoids are robust representations of the cluster centers, which is particularly important in the common context that many elements do not belong well to any cluster. The **`clara()`** function is similar, but is designed for larger data sets with say >200 observations.

Like K-means, the number of clusters used in PAM is fixed in advance, and an initial set of cluster centers is required to start the algorithm.

The function's usage is:

pam(x, k, diss = inherits(x, "dist"), metric = "euclidean", medoids = NULL, etc …)

The arguments are:

- **x** – data matrix or data frame, or dissimilarity matrix or object, depending on the value of the diss argument.
- **k** – positive integer specifying the number of clusters
- **diss** – logical flag: if TRUE (default for dist or dissimilarity objects), then x will be considered as a dissimilarity matrix. If FALSE, then x will be considered as a matrix of observations by variables.
- **metric** – character string specifying the metric to be used for calculating dissimilarities between observations. The currently available options are "euclidean" and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences. If x is already a dissimilarity matrix, then this argument will be ignored.

For additional documentation type "**?pam**" at the **R** prompt.

We can compute a robust K-means clustering around 8 medoids (clusters) by typing:

```
sitecl.pam<-pam(site.eucd,k=8)
sitecl.pam
```

Your results should look somewhat close to what appears below.

```
Medoids:
     ID
[1,] "24" "S24"
[2,] "26" "S26"
[3,] "31" "S31"
[4,] "28" "S28"
[5,] "35" "S35"
[6,] "12" "S12"
[7,] "14" "S14"
[8,] "16" "S16"
Clustering vector:
 S1  S2  S3  S4  S5  S6  S7  S8  S9 S10 S11 S12 S13 S14 S15 S16 S17 S18
  1   2   3   4   1   5   3   3   5   4   4   6   1   7   3   8   2   5
S19 S20 S21 S22 S23 S24 S25 S26 S27 S28 S29 S30 S31 S32 S33 S34 S35 S36
  1   4   1   2   2   1   1   2   5   4   1   3   3   3   3   1   5   2
S37 S38 S39 S40 S41 S42 S43 S44 S45
  1   4   5   1   4   4   5   4   1
Objective function:
   build     swap
1.868102 1.859847

Available components:
[1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
[6] "clusinfo"   "silinfo"    "diss"       "call"
```

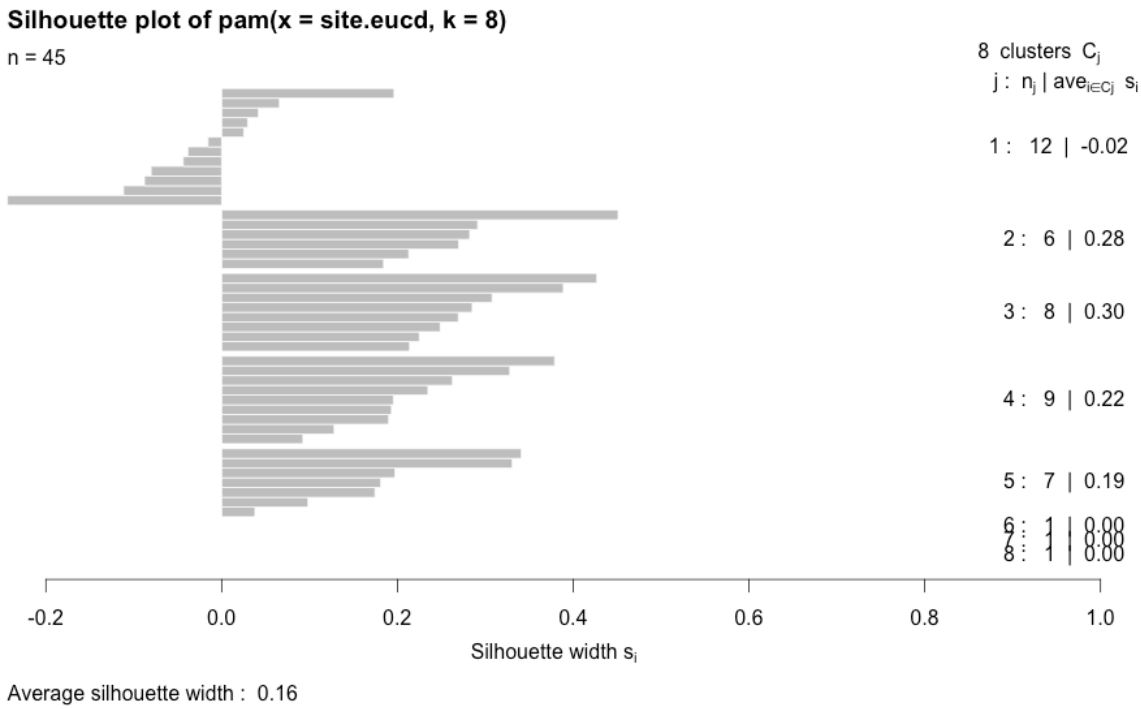
For large datasets, **`pam()`** may need too much memory or too much computation time. Then, **`clara()`** is preferable, see its documentation "**?clara**".

Once again you can extract specific results from the newly created object **`sitecl.pam`** by referencing the available components (this is true of any object in R). For example, you can obtain the site classifications by typing:

```
sitecl.pam$cluster
```

There is a special plotting function for outputs from the cluster library, including pam(). In R, just plot the pam object.

```
plot(sitecl.pam)
```



The plot is called a "Silhouette Plot", and shows for each cluster:

1. the number of objects per cluster = number of horizontal lines, also given in the right hand column
2. the means similarity of each object to its own cluster minus the mean similarity to the next most similar cluster (given by the length of the lines) with the mean in the right hand column
3. the average silhouette width

Objects which fit well within their cluster have a large positive Silhouette width; those which fit poorly have a small positive or even a negative Silhouette.

Importantly, you can produce the same Silhouette Plot for the K-means clustering above by typing:

```
plot(silhouette(sitecl.kmeans$cluster,site.eucd))
```

OPTIONAL READINGS

Belbin, L., and C. McDonald. 1993. Comparing three classification strategies for use in ecology. Journal of Vegetation Science 4:341-348.

Gauch, H.G. and R.H. Whittaker. 1981. Hierarchical classification of community data. Journal of Ecology 69: 537-557.

Hartigan, J.A. and M.A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society, Series C 28: 100–108.

Kaufman, L.A. and P.J. Rousseeuw. 1990. Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, New York, NY, US.

EXERCISE

Purpose

In this exercise, you will explore non-hierarchical approaches to cluster analysis.

Tasks

- Perform k-means clustering on your own data (or the class data) and analysis the results.
    - What type of cluster diagnostics should you perform?
    - How did you determine the “optimal” number of clusters?
- Perform a Partitioning around medoids using the class species abundance dataset and compare the results to a Partitioning around medoids based on the environmental data set.
    - How do these compare? That is, do sites that cluster according to similarities based on environment also cluster according to similarities based on species composition?

6. Principal Component Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Perform principal component analysis (PCA)

R Packages: vegan, FactoMineR, factoextra, dplyr
R Source: biostats

BACKGROUND

Ordination or scaling methods achieve an efficient and optimized low-dimensional representation of a complex data structure by emphasizing and bringing to the forefront underlying trended variation while suppressing "noise" (Kenkel 2006). This objective is readily achievable since multiple variables normally show coordinated responses to one or more (typically many) underlying factors. Ordination methods are generally used in exploratory data analysis, both to search for and summarize underlying trends, and to examine interrelationships among variables. A number of ordination approaches are available, including principal component analysis (PCA) and its variants, correspondence analysis (CA), and non-metric multidimensional scaling (NMDS). We will cover these approaches and others in the following chapters.

Principal component analysis (PCA) is an ordination technique that reduces the number of variables in an object-by-descriptor variable matrix into a fewer number of synthetic dimensions that are linear combinations, or principal components (PC), of the original variables. Put another way, PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first axis maximizes linear variance (i.e., it summarizes the dominant linear trend), the second maximizes the residual variance not accounted for by the first axis (i.e., subdominant linear trend), and so forth. The method is somewhat analogous to simple linear regression, but extended to multiple dimensions and without distinguishing between dependent and independent variables. In general, the aim is to represent the objects (rows) and variables (columns) of the data set in a new system of coordinates in reduced space (e.g., two or three axes or dimensions). The resulting ordination plot (usually depicted using the first 2 or 3 PCs) is used to interpret object similarity: objects that are close in ordinate space share similar properties and those distant are increasingly dissimilar. Conceptually, ordination complements cluster analysis in that they both aid interpretation and highlight structure underlying object similarity using multivariate data. However, PCA also allows one to investigate which variables are the primary drivers of object dissimilarity by examining the loadings (correlations) of the original variables against each PC.

In practice, PCA is either performed on a variance–covariance matrix or on a correlation matrix. The *variance-covariance approach* is followed when the same units or data types are used. The aim is then to preserve and to represent the relative positions of the objects and the magnitude of variation between variables in the reduced space. By contrast, PCA on a *correlation matrix* is used when descriptor variables are measured in different units or on different scales (e.g. different environmental parameters) or when the aim is to display the correlations among (standardized) descriptor variables. The two approaches lead to different principal components and different distances between projected objects in the ordination; hence, the interpretation of the relationships must be made with care. Indeed, for correlation matrices, variables are first standardized (i.e. they become independent of their original scales), and so distances between objects are also independent from the scales of the original variables. All variables thus contribute to the same extent to the ordination of objects, regardless of their original variance.

The mathematical basis of PCA is similar to multiple regression analysis, and correspondingly, requires that the original variables meet basic assumptions of multivariate normality, homogeneity in variance

and several others which are documented elsewhere (Legendre and Legendre 1998) and discussed previously in class. As an aside, these assumptions are generally violated by species abundance data that are often Poisson or log-normally distributed. Transformation of species abundances can sometimes abate violations of normality, but PCA also operates under the assumption that the original variables are represented in Euclidean distance. The high frequency of zeros that are typical of species community datasets makes this a dubious assumption. The resulting ordination space may be distorted and provide little ecologically relevant information on object similarity when using species data. On the other hand, continuous environmental data such as temperature, salinity, etc., are normally distributed (or transformations thereof) and are ideal for PCA.

SET-UP

In this exercise you will be working with the MAHA environment dataset. Import the dataset by typing:

```
envdata <- read.csv('MAHA_environment.csv', header=TRUE, row.names=1)
```

For exploratory purposes, we will also be using the MAHA site group dataset that contains three column vectors depicting the major basin, degree of human disturbance and a column of text colors (which will help for graphing later) for each site (object). Import the dataset by typing:

```
sitegroup <- read.csv('MAHA_Groups.csv',header=TRUE, row.names=1)
```

Let's first set-up your R work session by defining the current work directory to your folder of choice and loading the Packages listed at the beginning of this chapter. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

PRINCIPAL COMPONENT ANALYSIS

There are two basic functions for performing PCA in R: princomp() and prcomp(). Essentially, they compute the same values (technically, princomp computes an eigenanalysis and prcomp computes a singular value decomposition). We will make use of the prcomp () function.

Conduct the PCA

The prcomp() function accepts matrices or data frames that contain only the samples and variables of interest. Alternatively, you can select the columns from a larger matrix to include the analysis. The prcomp() function allows us to specify a number of parameters concerning the calculation. The most important is whether we want to use a correlation or variance-covariance matrix (see lecture and discussion above). To reiterate, PCA is sensitive to the scale of measurement of the data. If all the data are not measured on the same scale, using covariance means that the result will be determined mostly by the variable with the largest values, as it will have the highest variance. Using a correlation matrix treats all variables the same (standardized to mean=0 and std. dev.=1). In prcomp(), this means specifying scale=TRUE in the function call.

The usage of prcomp() is:

prcomp(x, scale=FALSE, scores=TRUE)

Where

- **`x`** is the object×variable matrix
- **`scale`** is a logical value. If TRUE, the original variables are scaled to unit variance (this is the correlation matrix and is almost always preferred). Default is FALSE and therefore the variance-covariance matrix is used in the PCA.

- **scores** is a logical value. If TRUE, PC scores are retained in the resulting PCA object. Default is TRUE.

For additional documentation of all the arguments type **?prcomp**.

Let's perform PCA by typing:

```
env.pca<-prcomp(envdata, scale=TRUE)
```

Interpreting the results from the PCA

The results of a PCA are complex and stored in a list, including:
- variance (eigenvalue) explained by each eigenvector
- variable loadings by eigenvector
- sample scores and ordination plots

First, let's evaluate how much variance in the dataset was captured by the resulting principal components (PCs). We do this by calling the summary of the object env.pca to view the variance explained by each PC and the cumulative variation explained sequentially (the results for PC 9 and 10 are omitted for clarity). Do this by typing:

```
summary(env.pca)
```

```
Importance of components:
                          PC1   PC2   PC3   PC4    PC5    PC6    PC7    PC8
Standard deviation     1.655 1.377 1.203 1.168 0.8916 0.7472 0.7225 0.5870
Proportion of Variance 0.274 0.190 0.145 0.136 0.0795 0.0558 0.0522 0.0345
Cumulative Proportion  0.274 0.463 0.608 0.745 0.8241 0.8799 0.9321 0.9666
```

Note that this summary presents the standard deviations for each PC and not the variances (i.e., the eigenvalues). Therefore, technically you have to square these values to obtain the eigenvalues. You can do this by calling the values from the env.pca object (recall you can type names(env.pca) to see all available results) by typing:

```
env.eig<-env.pca$sdev^2
env.eig
```

```
[1] 2.73922454 1.89544374 1.44731905 1.36396653 0.79503350 0.55833586
[7] 0.52197264 0.34456540 0.32123262 0.01290611
```

Alternatively you can use function pca.eigenval() which is contained in the BIOSTATS file (remember to source this at the beginning of your session!). Try typing (first 5 PCs are presented for clarity):

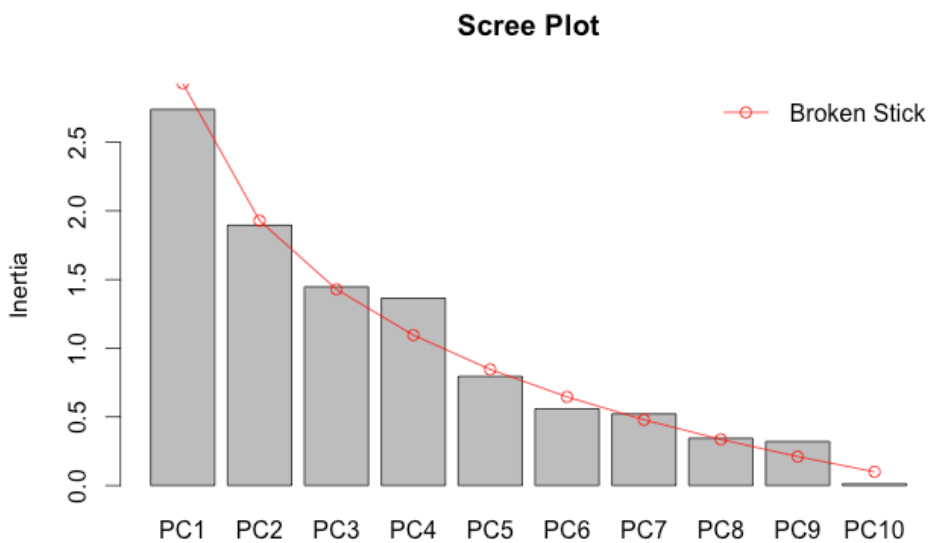
```
pca.eigenval(env.pca)
```

```
Importance of components:
                             PC1       PC2       PC3       PC4       PC5
Variance(eigenvalue)   2.7392245 1.8954437 1.4473191 1.3639665 0.7950335
Proportion of Variance 0.2739225 0.1895444 0.1447319 0.1363967 0.0795033
Cumulative Proportion  0.2739225 0.4634668 0.6081987 0.7445954 0.8240987
Broken-stick value     2.9289683 1.9289683 1.4289683 1.0956349 0.8456349
```

Notice that the values that you calculated in env.eig are identical to the summary table above.

Next, we can create a screeplot where each eigenvalue for each successive PC is depicted. If no correlation between variables existed in the original data set, the resulting PCs would be non-significant. That is, if the original variables are orthogonal, then it is impossible to collapse the variables into a lower dimensional space and the resulting PCs would be spurious. To determine whether our resulting eigenvalues are higher than expected purely by chance, we can overlay a null distribution of eigenvalues derived from the broken-stick model (these values are also presented in the summary table above); depicted by the line. In the plot below "Inertia" refers to the eigenvalue; depicted by the bars. Eigenvalues greater than the broken-stick expectation are considered to explain a statistically significance proportion of the variance in the original dataset.

```
screeplot(env.pca, bstick=TRUE)
```



From the above table and figure we can see that a large proportion of the variance in the dataset is explained by the first two PCs (46%). However, both PCs are below the null broken stick values, indicating that the variance explained is not a significant proportion of the total variance. However, ecological significance is very different from statistical significance, and I would argue that the PCA (and hence the ordination plot) is doing a good job by representing close to half of the total variation in the original data set.

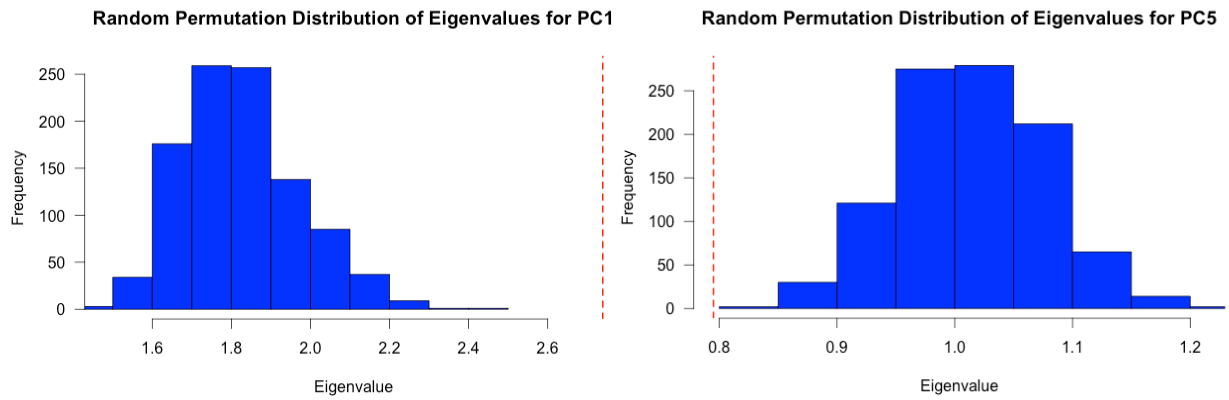
Alternatively, we may wish to test the statistical significance of the first several eigenvalues using a Monte Carlo (randomization) test. Below we conduct this analysis for the first 5 PCs (i.e., dimensions) to reduce computing time, by typing:

```
ordi.monte(envdata,ord='pca',dim=5)
```

This test conducts, by default, 1000 random permutations of the data matrix, permuting each column independently to remove any real correlation structure among variables, and compares the original eigenvalue to the distribution of eigenvalues under the null hypothesis of no real correlation structure. Eigenvalues with p-values less than some critical value (e.g., 0.05) are considered "significant." The random distributions of eigenvalues for each PC will be plotted, in addition to a summary table. Below are the plots for the first and last PCs.

```
Randomization Test of Eigenvalues:
              PC1    PC2    PC3    PC4    PC5
```

```
Eigenvalue 2.739 1.895 1.447 1.364 0.795
P-value    0.000 0.000 0.084 0.000 1.000
```



Eigenvectors (variable loadings):

We can examine the eigenvectors (i.e., variable loadings) on each PC (only PC1-5 are presented), by typing:

```
env.pca$rotation
```

```
                  PC1        PC2         PC3         PC4         PC5
Sinuosity  0.00429546 -0.4735237 -0.22940933  0.27361829 -0.47648306
Slope      0.26319558  0.1303184 -0.34968154 -0.46323791 -0.25803639
WDRatio    0.37836949 -0.1633701  0.31612993  0.23288139  0.30258611
SubEmbed  -0.32966590  0.2478083  0.33469255  0.25747281 -0.39866023
HabQual    0.35595505 -0.3516462 -0.24667911 -0.05692156 -0.31449155
Elev       0.23866884 -0.2113364  0.35123065 -0.53998940  0.19735782
RoadDen   -0.26905123 -0.2427914 -0.49904214  0.15497594  0.52202538
Agricult  -0.41161178 -0.2821745  0.25449588 -0.38762312 -0.18790521
HumanUse  -0.48328053 -0.3592271 -0.02179364 -0.23257076  0.10093767
BasinAre   0.14080781 -0.4845218  0.33858118  0.25851631  0.00551955
```

The loadings indicate the correlation of the original variables with each PC. Strong correlations (either positive or negative) indicate a high contribution towards the linear combination comprising each PC.

Alternatively, we can use the pca.eigenvec() function as follows:

```
pca.eigenvec(env.pca,dim=5,digits=3,cutoff=.1)
```

This function suppresses small values below the specified cutoff value (the default is 0, so all coefficients are printed) to emphasize the more important ones. Recall that eigenvector coefficients derived from the correlation matrix are proportional to the correlations between the original variables and the ordination axes, but they are NOT correlation coefficients. Nevertheless, large positive values indicate that samples positioned on the positive end of the ordination axis contain larger values of the corresponding variable, and vice versa for negative values. The dim=3 means that only the first three components are listed.

It may be more useful and interpretable to convert the eigenvector coefficients into simple correlation coefficients (i.e., Pearson product-moment correlations), as follows:

```
pca.structure(env.pca,envdata,dim=5,cutoff=.4)
```

These structure coefficients (also referred to as structure correlations, correlation loadings, or just loadings) are actual simple linear correlations between the original variables and the principal component scores. The squared structure coefficients give the percentage of variance in each original variable accounted for by each principal component.

The resulting output would look like:

```
Structure Correlations:
             PC1    PC2    PC3    PC4    PC5
Sinuosity        -0.652               -0.425
Slope      0.436        -0.421 -0.541
WDRatio    0.626
SubEmbed  -0.546         0.403
HabQual    0.589 -0.484
Elev                     0.423 -0.631
RoadDen   -0.445          -0.6         0.465
Agricult  -0.681               -0.453
HumanUse    -0.8 -0.495
BasinAre         -0.667  0.407
```

Based on the eigenvectors or the structure coefficients we generate an ecological interpretation of each principal component.

<u>Sample scores and ordination plots:</u>

Each sample has a score or location on each principal component axis. To see these sample scores, which are stored in the component named 'x' in the list object output from the prcomp() function, simply type:

```
env.scores<-env.pca$x
env.scores
```

```
            PC1          PC2           PC3          PC4           PC5          PC6
S1  -1.89423579  0.44448692  0.529427773  0.24783004  0.486832947 -0.06903318
S2  -2.45393510  2.51644089  0.514761313  0.98848294 -0.440977886  0.09479069
S3  -2.05646139 -0.70986569 -0.172173175 -1.06576107  0.117445919 -0.37566850
S4   0.80053906  1.50030939 -0.648110305 -0.06331708 -0.843362669  0.36315213
S5   0.44939256  0.75494057 -0.746555367  0.31624254 -0.982817687  0.91857748
S6   0.33102775 -2.04471858  0.786058382  0.93457905 -0.340067913  1.41184369
```

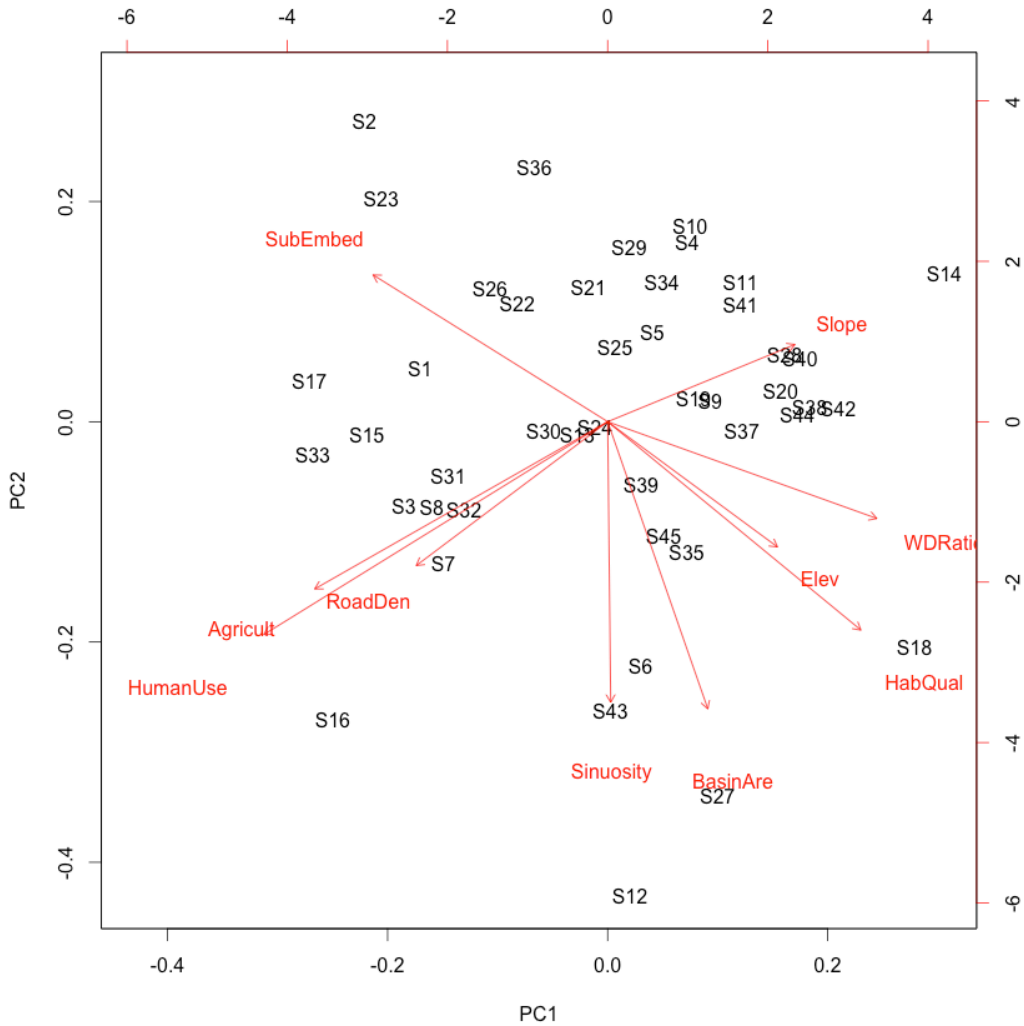
These scores are the standardized scores for each sample on each principal component axis; that is, they represent the position of each sample on each standardized principal component axis. Recall from lecture that these scores can be obtained directly by multiplying the eigenvector coefficients by the corresponding values of the standardized data for each variable and summing.

Next, we can visualize the ordination using biplot() or ordiplot().

First, let's take the easy approach and type:

```
biplot(env.pca)
```

You will produce the following ordination biplot.



In the above figure, the vector arrows indicate the direction of variable gradients in ordination space. The longer the arrow the stronger, or more important, the respective variable is for describing the PCs. In this example, sites in the lower left corner (third quadrant) of the biplot have high values for the variables HumanUse, Agricult and RoadDen and lower values for variables with vector arrows that point towards the upper right corner (arrows that are at an increasingly oblique angle to the third quadrant). Sites located in close proximity in ordination space (e.g., S7, S32 and S31) share similar environmental features (in this case, they are all characterized by high percentages of upstream watersheds in agricultural or human land use and high road densities), and sites located far apart (e.g., S12 and S2) are very different environmentally.

Additional visualization options

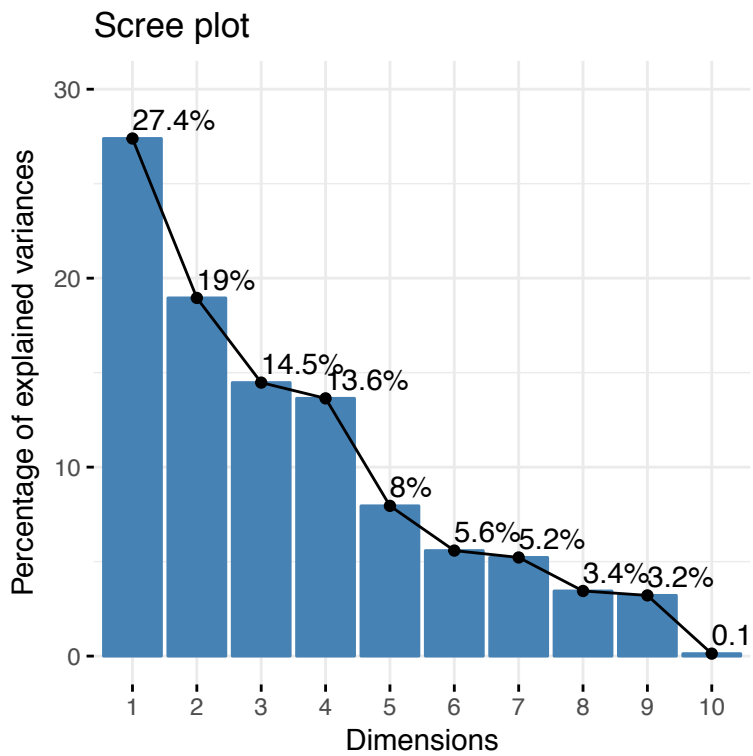
Let's use the FactoMineR and factoextra packages to produce some nicer looking outputs from PCA. In a nutshell, factoextra is an **R** package to easily extract and visualize the output of multivariate data analyses conducted in FactoMineR.

First type the following to conduct the PCA:

```
env2.pca<-PCA(envdata,graph=F)
```

Type the following to produce the scree-plot of eigenvalues:

```
fviz_screeplot(env2.pca, addlabels = TRUE, ylim = c(0, 30))
```

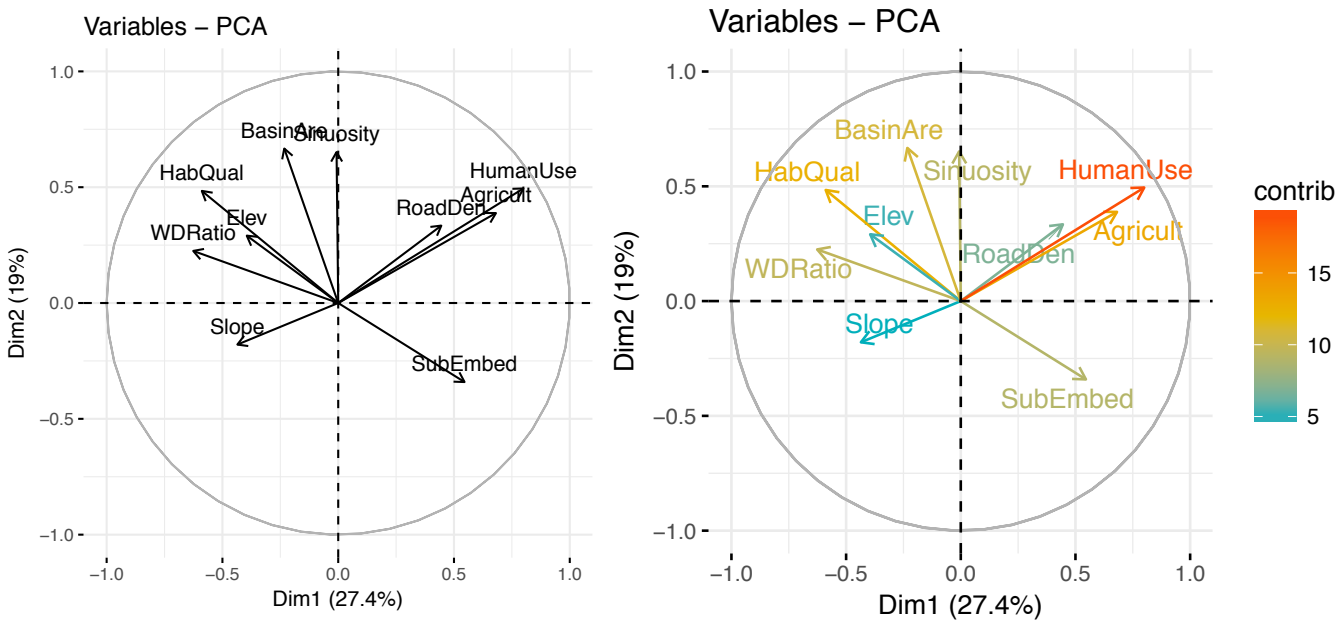


Now, let's look at the variable loadings (i.e., eigenvectors) by typing:

```
fviz_pca_var(env2.pca, col.var = "black")
```

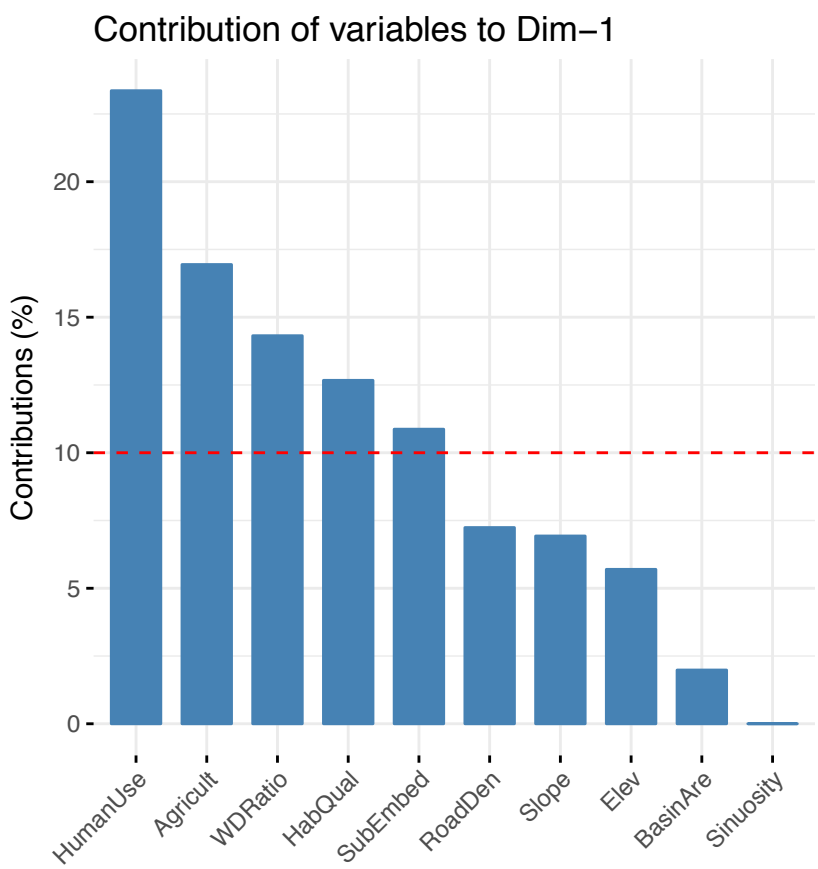
and a fancy looking one where total contributions of the variables are color coded:

```
fviz_pca_var(env2.pca, col.var="contrib",gradient.cols = c("#00AFBB",
"#E7B800", "#FC4E07"), repel = TRUE)
```



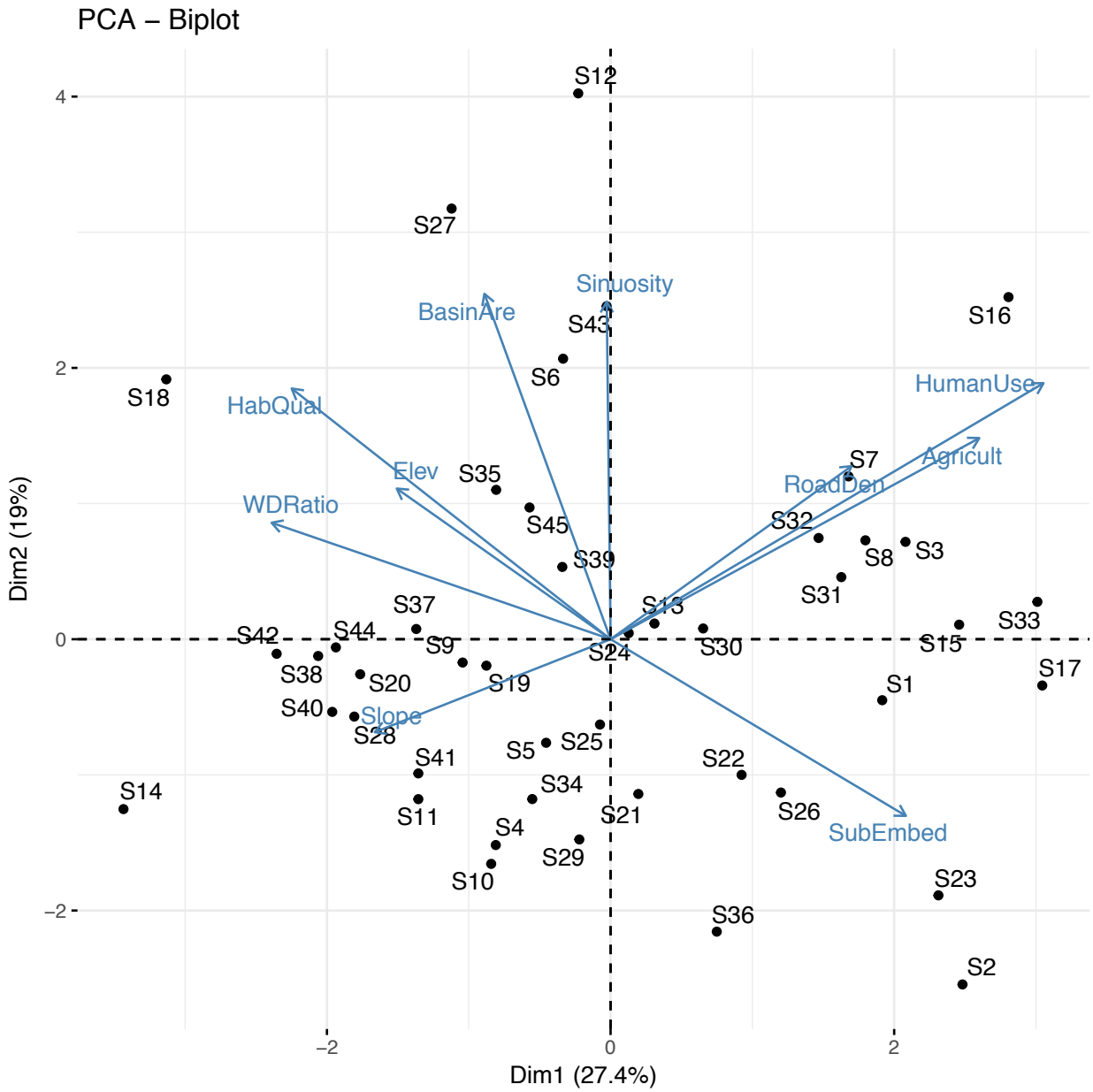
Now let's look at the relative contributions of the variables to the first PC, by typing:

```
fviz_contrib(env2.pca, choice = "var", axes = 1, top = 10)
```



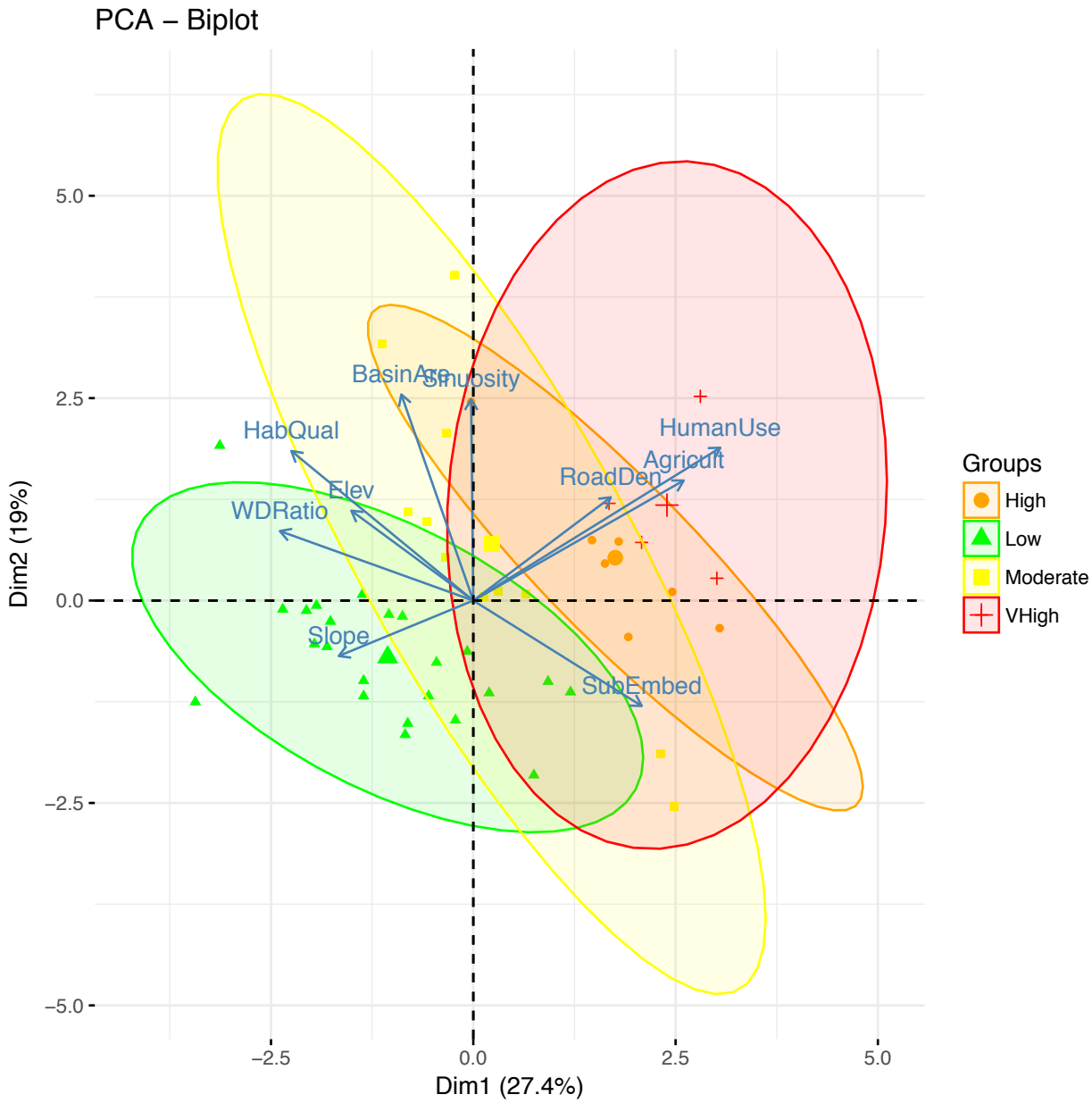
Producethe PCA biplot by typing:

```
fviz_pca_biplot(env2.pca, repel = TRUE)
```



Finally, let's now interpret the ordination plot by symbolizing each site (point) according to their level of human disturbance contained in the sitegroup dataset. We will also draw concentration ellispes. Try typing:

```
fviz_pca(env2.pca,label = "var", habillage = sitegroup$DISTURB,palette =
c("orange", "green", "yellow","red"), addEllipses = TRUE)
```



Here, the larger symbols for each disturbance category represents the centroid. You can remove the eigenvectors by using the following code:

```
fviz_pca_ind(env2.pca,habillage = sitegroup$DISTURB,palette = c("orange",
"green", "yellow","red"), addEllipses = TRUE)
```

Also, I should you mention that you can use ordiplot () in the vegan library to plot the results. Its usage is:

ordiplot(ord, choices = c(1, 2), type="points", display, xlim, ylim, ...)

Where

- **ord** is the result from an ordination
- **choices** are the axes shown
- **type** is the type of graph which may be "points", "text" or "none" for any ordination method.
- **display** only "sites" or "species". The default for most methods is to display both.
- **xlim, ylim** the x and y limits (min,max) of the plot.
- **labels** is optional text used for labels. Row names will be used if this is missing.

For additional documentation of all the arguments type **`?ordiplot`** in **R**.

If you want, you can try typing:

```
ordiplot(env.pca, choices = c(1, 2), type="text", display="sites", xlab="PC 1
(27%)", ylab="PC 2 (18%)")
```

Next, we can add the eigenvectors (i.e., the variable loadings) on the ordination, using the following command:

```
arrows(0,0,env.pca$rotation[,1]*5,env.pca$rotation[,2]*5,col='purple')

text(env.pca$rotation[,1]*5.2,env.pca$rotation[,2]*5.2,row.names(env.pca$rota
tion))
```

Note that we multiplied the eigenvectors by a constant (in this case, 5) so that the vectors would not be too small to interpret. Then we added the text labels to the end of the arrows.

OPTIONAL READINGS (* recommended)

Rose, C. and Smith, M. D. 1996. The Multivariate Normal Distribution. Mathematica J. 6: 32-37.

Gauch, H.G., and R.H. Whittaker. 1972. Comparison of ordination techniques. Ecology 53: 868-875.

Goodall, D.W. 1954. Objective methods for the classification of vegetation. III. An essay in the use of factor analysis. Australian Journal of Botany 2: 304-324.

James, F.C. and McCulloch. 1990. Multivariate analysis in ecology and systematics: Panacea or Pandora's box. Annual Review in Ecology and Systematics 21:129-166.

Pearson, K. 1901. On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2:559-572.

EXERCISE

Purpose

Upon completion of this chapter, you should be able to do the following: (1) Carry out a principal components analysis; (2) assess how many principal components should be considered in an analysis; (3) interpret principal component scores; (4) determine when a principal component analysis may be based on the variance-covariance matrix, and when the correlation matrix should be used; and (5) consider how principal component scores may be used in further analyses.

Tasks

- Perform a principal component analysis using the covariance matrix.
    - How many PCs are required to explain 90% of the total variation for this data? Are these statistical significant? How did you test for this?
    - For the number of components in part a, give a brief interpretation of each component.
    - Interpret the bi-plot (i.e., ordination with object scores and eigenvectors)
- Perform a principal component analysis using the correlation matrix. Repeat questions 1a-c.
- Conduct a PCA on the species occurrence dataset. What are your results and was this a good decision?

7. Principal Coordinate Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Principal coordinate analysis (PCoA)

R Packages: vegan, ape
R Source: biostats

BACKGROUND

Principal coordinate analysis (PCoA), also known as metric multidimensional scaling, is a generalized variant of PCA in that it uses a linear mapping of the distances between objects onto the ordination space, and the algorithm attempts to explain most of the variance in the original data set (Gower 1966). As opposed to PCA, PCoA allows for the ordination of data based on any association coefficient (not just correlation/covariance). Flexibility in the choice of similarity/distance matrix translates to flexibility in the application of PCoA. This is particularly useful for species and mixed data types, where PCA is not appropriate. Although there are important differences between PCoA and PCA, the eigenanalysis underpinning both approaches are very similar. For example, a PCoA ordination based on Euclidean distances among objects is identical to that of a PCA based on a covariance matrix among variables. Despite its usefulness, PCoA is rarely used by ecologists.

One minor drawback of PCoA is that doesn't provide a direct link between the components and the original variables and so the interpretation of variable contribution may be more difficult. This is because PCoA components, instead of being linear combinations of the original variables as in PCA, are complex functions of the original variables depending on the selected dissimilarity measure. Besides, the non-Euclidean nature of some distance measures does not allow for a full representation of the extracted variation into a Euclidean ordination space. In that case, the non-Euclidean variation cannot be represented and the percent of total variance cannot be computed with exactness. The choice of the dissimilarity measure is thus of great importance, and subsequent transformation of the data to correct for negative eigenvalues is sometimes necessary (see Legendre & Legendre, 1998, section 9.2.4. for how to correct for such negative eigenvalues). Although there is no direct, linear relationship between the components and the original variables, it is still possible to correlate object scores on the main axis (or axes) with the original variables to assess their contribution to the ordination.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**. Import the dataset by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

Let's transform the data before diving into the analysis. We will do this because the species abundance dataset is highly skewed and contains some rather large values which are valid but highly influential. Remember, the log of zero is undefined so we'll add 1 to each value in our data set.

```
speabu.log<-log(speabu+1)
```

PRINCIPAL COORDINATE ANALYSIS

Calculating the distance matrix

The first step in PCoA is the selection of a distance matrix to describe similarities among objects according to the descriptors. Again, any (dis) similarity coefficient can be selected. For today, let's generate a distance matrix using an appropriate dissimilarity coefficient for species abundance – Bray-Curtis coefficient. We'll use the vegdist() function. Its usage is:

vegdist (x, method, binary=FALSE)

where

- x is a data frame (here a sitexspecies abundance matrix)
- method is the desired similarity coefficient. Bray-Curtis ("bray"), Jaccard's ("jaccard"), Gower's ("gower") and several other similarity coefficients are available
- binary; logical. If equals TRUE, the original data set is converted to presence/absence (1, present; 0, absent) and then the desired similarity coefficient is applied. Default: FALSE.

For additional documentation type **?vegdist**.

Type,

```
speabu.d<-vegdist(speabu.log, "bray")
```

Perform the PCoA

Now we can perform PCoA. In vegan, the function is called cmdscale(),which refers to classic (metric) multidimensional scaling. Its usage is:

cmdscale(d, k, eig = FALSE, add = FALSE)

where

- d is a dissimilarity object (generated by dist or vegdist)
- k is the number of principal components (PC) that should be extracted from the distance matrix (max number = min(col, rows)-1)
- eig, logical. If TRUE eigenvalues for each PC are retuned. Default: FALSE.
- add, logical. If TRUE a constant is added to each value in the dissimilarity matrix so that the resulting eigenvalues are non-negative. Default: FALSE.

Let's perform PCoA:

```
spe.pcoa<-cmdscale(speabu.d, eig=TRUE, add=T)
```

Note that k is set to 35 (# species (columns) – 1). Type **`spe.pcoa`** at the prompt to view the returned principal scores (PCs) and eigenvalues.

The principal scores are contained in spe.pcoa$points and the eigenvalues are contained in spe.pcoa$eig.

The principal scores for the first 5 observations should read:

```
        [,1]        [,2]         [,3]         [,4]         [,5]
S1  -0.379962052 -0.29157310  0.008993613 -0.002743771  0.191458393
S2  -0.301474146 -0.29455879  0.388392767  0.011907596 -0.180354539
S3  -0.117052609 -0.22344133  0.044101350 -0.072273726  0.253420678
S4   0.662126194 -0.17361377 -0.208290288  0.155837150 -0.064838489
S5  -0.115646174 -0.31700403 -0.323314608  0.162097678 -0.006921289
```

The eigenvalues for the first should read:

```
[1] 5.2402590 3.6946927 3.1018646 2.3737229 2.0522328 1.6856022 1.5614293
[8] 1.3452143 1.1556406 1.1417494 1.0759328 1.0231588 0.9406274 0.8869959
[15] 0.7093863 0.6563471 0.6343436 0.5251099 0.4788888 0.4456985 0.4217005
[22] 0.4165729 0.3806084 0.3529408 0.3108155 0.2931376 0.2767940 0.2492598
[29] 0.2369490 0.2134872 0.2104035 0.1760313 0.1647904 0.1488193 0.1395068
```

The percent of variation explained by any principal coordinate can be calculated by dividing its eigenvalue by the sum of the eigenvalues across all PCs. Recall, the same approach is used in PCA. Let's calculate this for the first five PCs.

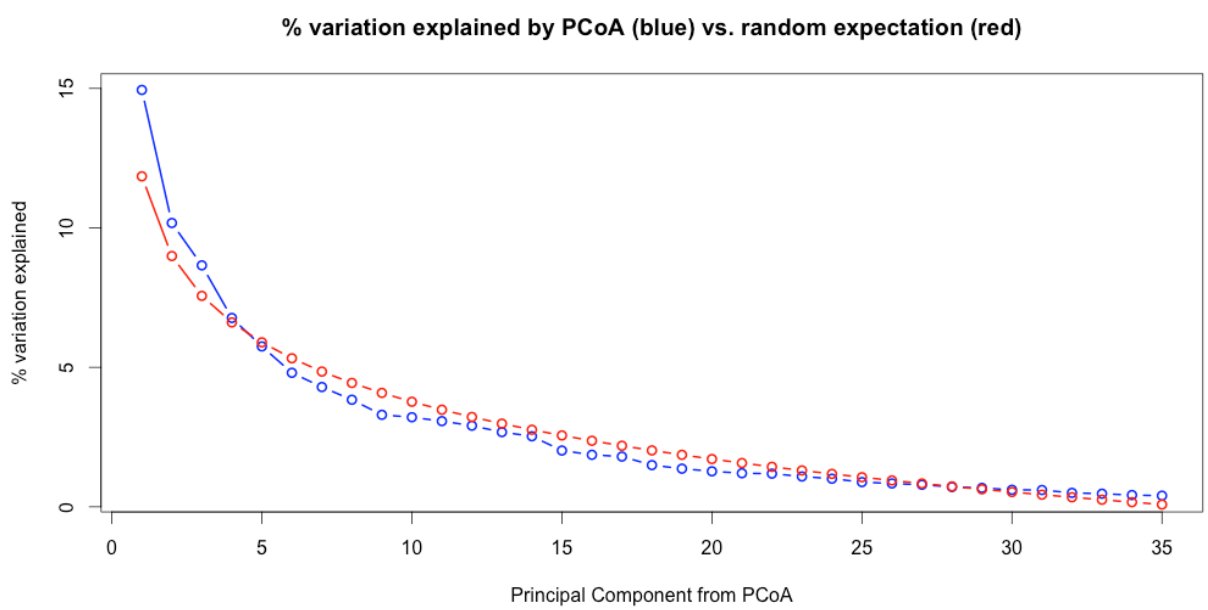
```
spe.pcoa$eig/sum(spe.pcoa$eig)*100
```

This should return the following values for the first 5 principal components:

```
[1]  1.494490e+01  1.017524e+01  8.655463e+00  6.769729e+00  5.749803e+00
```

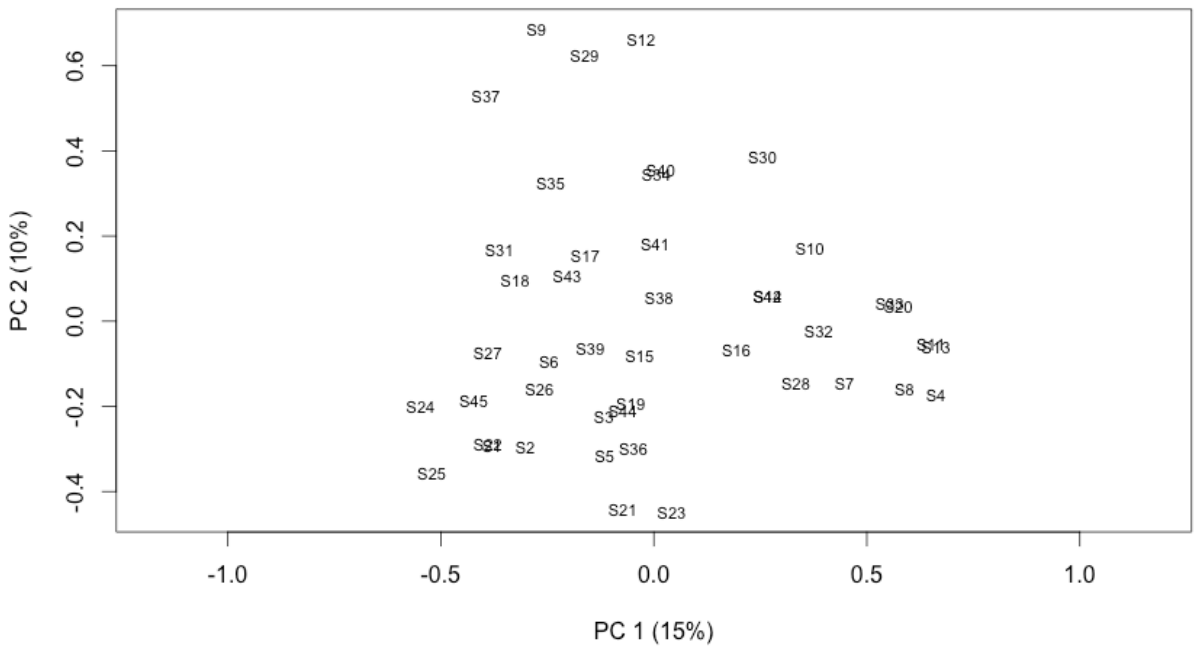
Let's compare the eigenvalues to expectations according to the broken stick model.

```
plot(spe.pcoa$eig[1:35]/sum(spe.pcoa$eig)*100,type="b",lwd=2,col="blue",xlab=
"Principal Component from PCoA", ylab="% variation explained", main="%
variation explained by PCoA (blue) vs. random expectation (red)")

lines(bstick(35)*100,type="b",lwd=2,col="red")
```



Now, let's view the ordination plot:

```
ordiplot(spe.pcoa, choices = c(1, 2), type="text", display="sites", xlab="PC
1 (15%)", ylab="PC 2 (11%)")
```



This plot represents each of the sites in 2-D ordination space (x-axis = principal component 1, y-axis = principal component 2).

Calculate the PC loadings (i.e., variable weights)

Now, to calculate and depict species loadings (i.e., principal weights in the eigenvectors) on each principal coordinate we'll use the function envfit() along with the PC scores from our PCoA object. For brevity, please see additional documentation and usage information in the vegan manual (type ?envfit, ?scores). The function envfit() simply performs a linear correlation analysis based on standardized data (in other words, a simple linear regression) between each of the original descriptors (i.e., species) and the scores from each principal component. A permutation test is used to assess statistical significance, rather than using the F distribution.

```
vec.sp<-envfit(spe.pcoa$points, k=45, speabu.log, perm=1000)
```

Another way of doing this is to extract the first two PCs from our PCoA object using the scores() function. To do this, type:

```
vec.sp<-envfit(scores(spe.pcoa), speabu.log, perm=1000)
```

This should return the values (note yours will be slightly different because it is based on random permutations) listed below for the first 7 species:

```
vec.sp
```

```
***VECTORS
```

```
              Dim1      Dim2     r2 Pr(>r)
BANDDART -0.183732  0.982976 0.3384 <0.001 ***
BANDSCUL -0.056155  0.998422 0.0598  0.262
BLACDACE  0.774004 -0.633181 0.6104 <0.001 ***
BLUECHUB -0.690920 -0.722931 0.7034 <0.001 ***
BLUEGILL -0.853874 -0.520480 0.1510  0.029 *
BLUNMINN -0.422624  0.906305 0.1471  0.025 *
BROOTROU  0.856519 -0.516115 0.0115  0.807
CCHUBSUC -0.740762 -0.671768 0.0412  0.451
COMMSHIN -0.971232  0.238134 0.1038  0.098 .
CREECHUB  0.932674  0.360719 0.0047  0.901
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
P values based on 1000 permutations.
```

Based on these results you will notice that banded darter (BANDDART), black dace (BLACDACE), blue chub (BLUECHUB), bluegill (BLUEGILL) and bluntnose minnow (BLUMINN) show statistically significant loadings on the first two principal components. These species could be used to interpret the position of the stream sites (objects) in ordination space. You can drill down into the sub-objects contained in vec.sp, by typing:

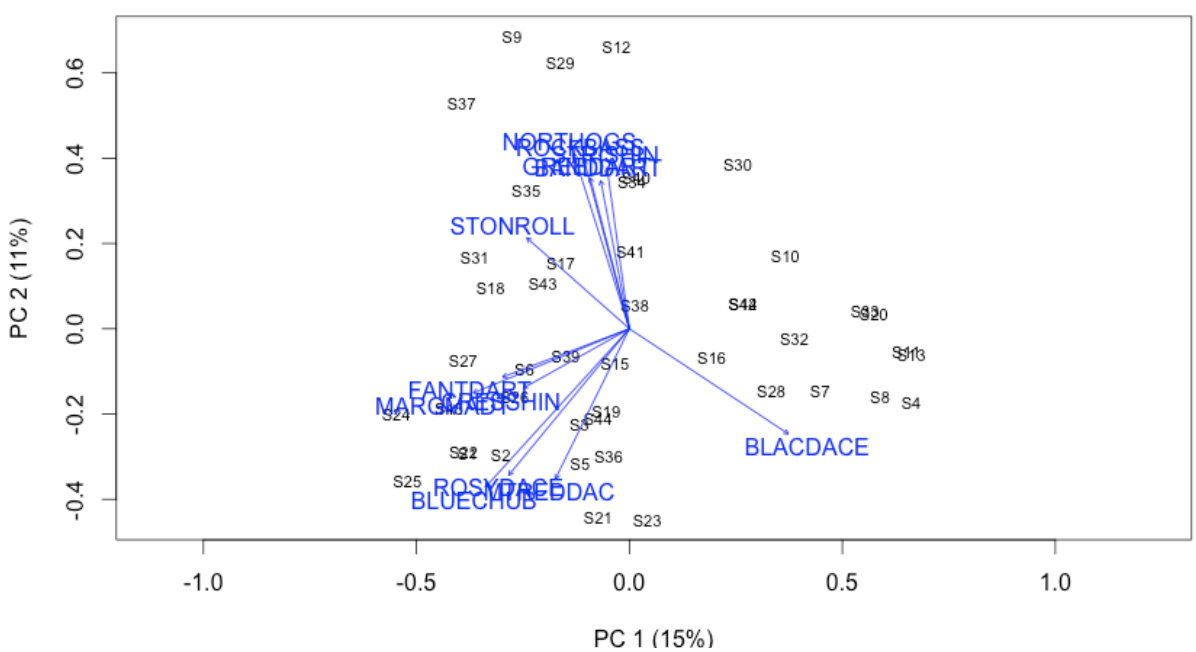
```
names(vec.sp)
names(vec.sp$vectors)
```

You will see that the normalized eigenvectors (scaled to unit length) are presented in the matrix vec.sp$vectors$arrows. These values should look familiar!

Now, plot the eigenvectors on the ordination plot.

```
ordiplot(spe.pcoa, choices = c(1, 2), type="text", display="sites", xlab="PC 1 (15%)", ylab="PC 2 (11%)")
plot(vec.sp, p.max=.01, col="blue")
```

where p.max is the significance level that the species occurrence data must have with either PC in order to be depicted (these p-values were presented in vec.sp).



Other options for conducting PCoA

It is common that multiple functions exist to perform the same multivariate analysis, and this is the case for PCoA. You can use the pcoa() function in the ape package. Be sure to install the library first.

Here is some example code:

```
spe.pcoa<-pcoa(speabu.d,correction="cailliez")
```

OPTIONAL READINGS

Gower JC (1966) Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53: 325–338.

Podani, J. 2005. Multivariate exploratory analysis of ordinal data in ecology: pitfalls, problems and solutions. J. Veg. Sci. 16: 497–510.

Podani, J. and Miklos, I. 2002. Resemblance coefficients and the horseshoe effect in principal coordinates analysis. Ecology 83: 3331–3343.

8. Non-metric Multidimensional Scaling

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Non-metric Multidimensional scaling (NMDS)

R Packages: vegan, cluster
R Source: biostats

BACKGROUND

Principal component analysis (and its variants) utilizes matrix algebra to derive unique, successive linear axes such that distances among objects in multivariate space are well represented in a low-dimensional ordination space. Non-metric multidimensional scaling (NMDS) obtains a similar representation using only the rank order of distances, rather than the distance values themselves (Digby and Kempton 1987). The theoretical advantage of this rank-order approach to ordination is that underlying assumptions of linearity (as in PCA and variants) are not required nor specified. The lack of underlying assumptions, however, necessitates the use of a computationally intensive iterative algorithm to derive an optimized ordination configuration. At each NMDS iteration the rank order relationship between ordination and variable space distances is improved through successive approximation. Iteration continues until the stress function, which measures the correspondence between ranked ordination and multivariate space distances, is minimized. In other words, the *stress* value indicates the faithfulness of the ordination configuration to the original dissimilarity matrix. The final solution is an optimized rank-order mapping of the sampling units in an ordination space of specified dimensionality. The solution obtained from a single run may not be globally optimal, however, since NMDS is based on an iterative algorithm. It is therefore imperative that multiple NMDS solutions be obtained in order to ensure that a stable and optimal ordination configuration is found. Because only rank order relationships are used, NMDS solutions are unstable or even degenerate when applied to small data sets or to poorly structured data.

Previously, the numerically intensive calculations required to achieve a stable NMDS solution precluded its use in ecology. But with increased computing power this approach has grown in popularity and is now arguably one of the most commonly used ordination techniques in community ecology. Part of this popularity is owed to NMDS's ability to handle any (dis)similarity coefficient. For species abundance data sets, simulation testing has indicated that NMDS may outperform other commonly used ordination techniques (e.g., PCA, PCoA, CA, DCA; Gauch 1982, Reynolds et al. 1988, McCune 2002). That is, the object configuration depicted in the NMDS ordination better captured the simulated species abundance patterns.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**. Import the dataset by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
```

Let's transform the data before diving into the analysis. We will do this because the species abundance dataset is highly skewed and contains some rather large values which are valid but highly influential. Remember, the log of zero is undefined so we'll add 1 to each value in our data set.

```
speabu.log<-log(speabu+1)
```

NMDS

To perform NMDS we'll use the function metaMDS() which is part of the vegan library. Its usage is:

metaMDS(comm, distance = "bray", k = 2, trymax = 20, autotransform =TRUE)

Where:

1. comm is the community data set (object-by-descriptor matrix)
2. distance is the dissimilarity coefficient used (calculated using vegdist). Default is Bray-Curtis.
3. k, the number of ordination axis to generate
4. autotransformation, logical. If TRUE, metaMDS autotransfoms the data (usually using square root transformation) using simple heurisitics. Default is TRUE, but set this to FALSE.
5. trymax, the maximum number of random starts in search for stable solution. Default is 20.

metaMDS combines several functions from isoMDS (in the MASS package) that automate the NMDS iterative process. To perform NMDS piecemeal, you can type ?metaMDS for additional documentation of the underlying functions. Let's perform metaMDS using our log-transformed dataset.

```
spe.nmds<-metaMDS(speabu.log, distance='bray', k=2, autotransform=FALSE,
trymax=100)
```

Type **spe.nmds**. Your results should be somewhat similar as below.

```
Call:
metaMDS(comm = speabu.log, k = 2, trymax = 100, autotransform = FALSE)

Nonmetric Multidimensional Scaling using isoMDS (MASS package)

Data:     speabu.log
Distance: bray

Dimensions: 2
Stress:     0.2320339
Two convergent solutions found after 36 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'speabu.log'
```

You can type **names(spe.nmds)** to obtain a list of objects resulting from the analysis.

```
[1] "points"    "dims"      "stress"    "data"      "distance"  "converged"
[7] "tries"     "species"   "call"
```

For example, spe.nmds$points contains the co-ordinates from the first two axes (in case you wanted to plot the results in another program). From the above summary we can see that the stress level is relatively high and therefore indicates a poor fit between the original distance matrix and the final ordination configuration. To improve the fit, we could try a different transformation (presence/absence) or perform NMDS with three axis (k=3). We'll repeat the procedure here.

```
spe.nmds2<-metaMDS(speabu.log, distance='bray', k=3, autotransform=FALSE,
trymax=100)
```

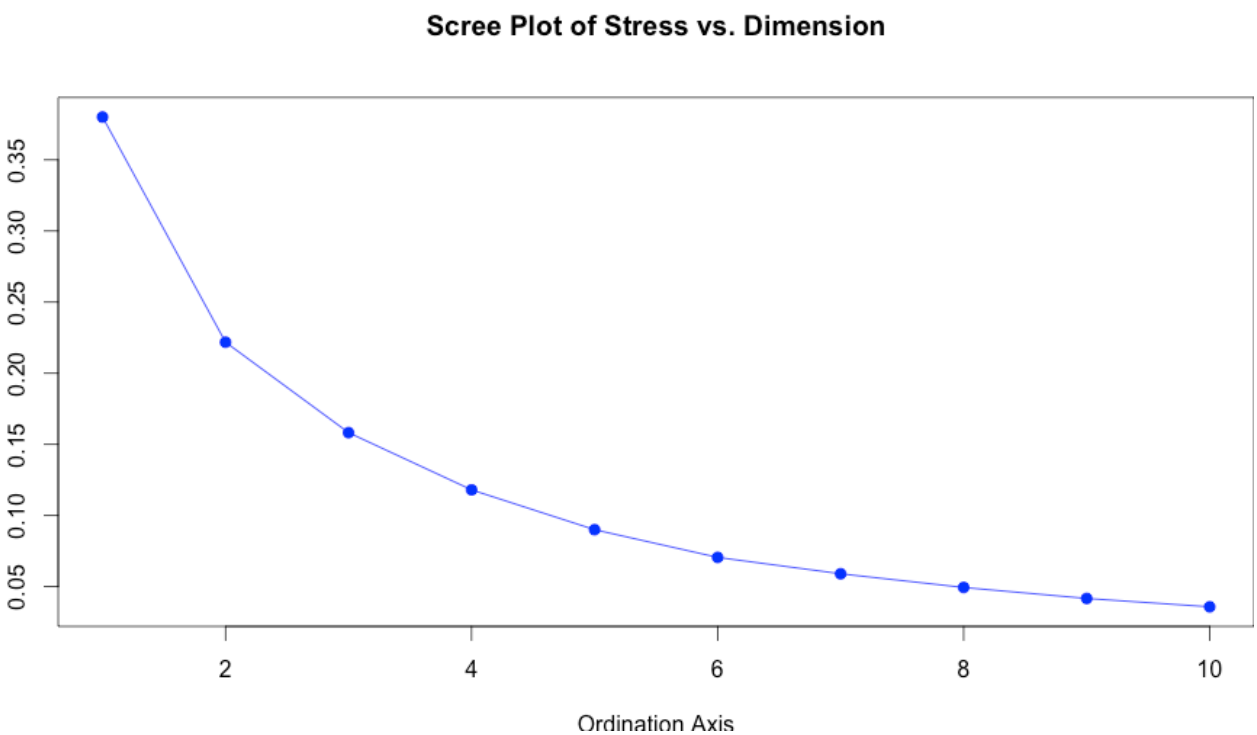
On this run, the stress improved to 16.4 indicating that the major gradients in the data set can be sufficiently captured by three dimensions. Of course, increasing the number of ordination axis will

reduce the stress value (at k = n species, stress must = 0 because all species would be represented by a unique ordination axis), but the utility of NMDS, as with any ordination technique, is to summarize as much variation as possible using the fewest number of axis. Ultimately, it is the user who must decide whether the addition of dimensions is justified by the reduction in stress.

Examining a scree plot of stress versus the number of dimensions can help you make this decision. To perform this type:

```
nmds.scree(speabu.log, distance='bray', k=10, autotransform=FALSE, trymax=20)
```

You will be provided a scree plot that looks similar to this:



This function basically calls the metaMDS function as before, but this time it calls it once for each number of dimensions and then plots the final stress value against the number of dimensions.

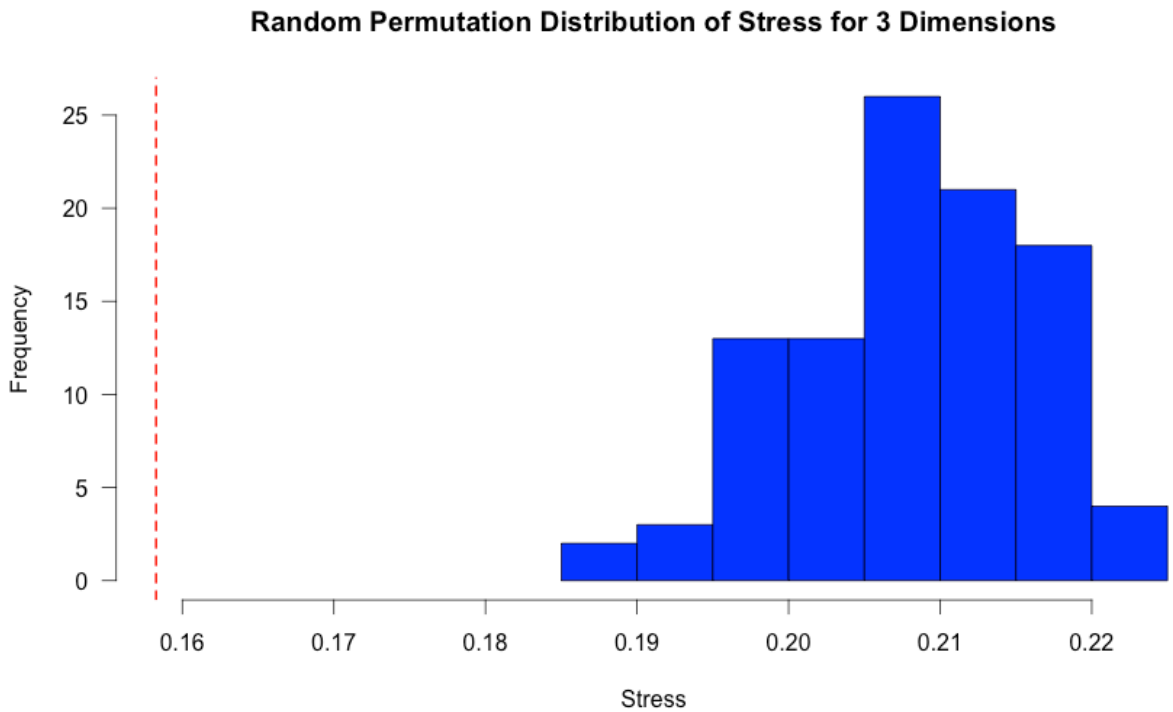
Once the final number of dimensions has been decided upon, a Monte Carlo randomization test of the final stress value can be conducted as follows. Note that this will take a couple minutes to complete.

```
nmds.monte(speabu.log, distance='bray', k=3, autotransform=FALSE, trymax=20)
```

This will return the permuted stress values (and histogram) and calculated p-value.

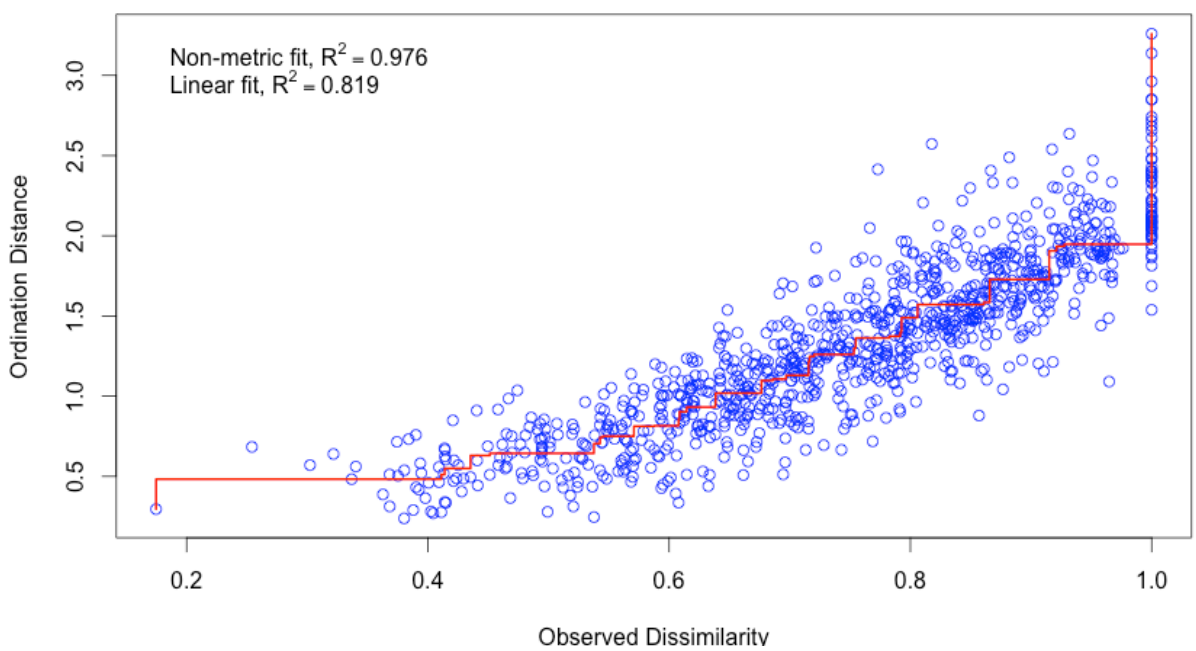
```
Randomization Test of Stress:
Permutation stress values:
  [1] 20.81755 21.27511 20.59101 21.77180 21.60949 20.86109 21.47368 19.93579
  [9] 21.10165 21.90203 21.65971 20.55388 21.27061 20.55968 21.74566 20.94443
 [17] 21.62887 21.04472 21.32917 21.90709 20.66803 20.82322 19.69421 20.85727
 [25] 21.79995 20.46723 20.99515 21.60599 20.84756 21.62835 20.23694 20.29475
…
```

```
      [,1]
Observed stress 16.36179378
P-value          0.00990099
```



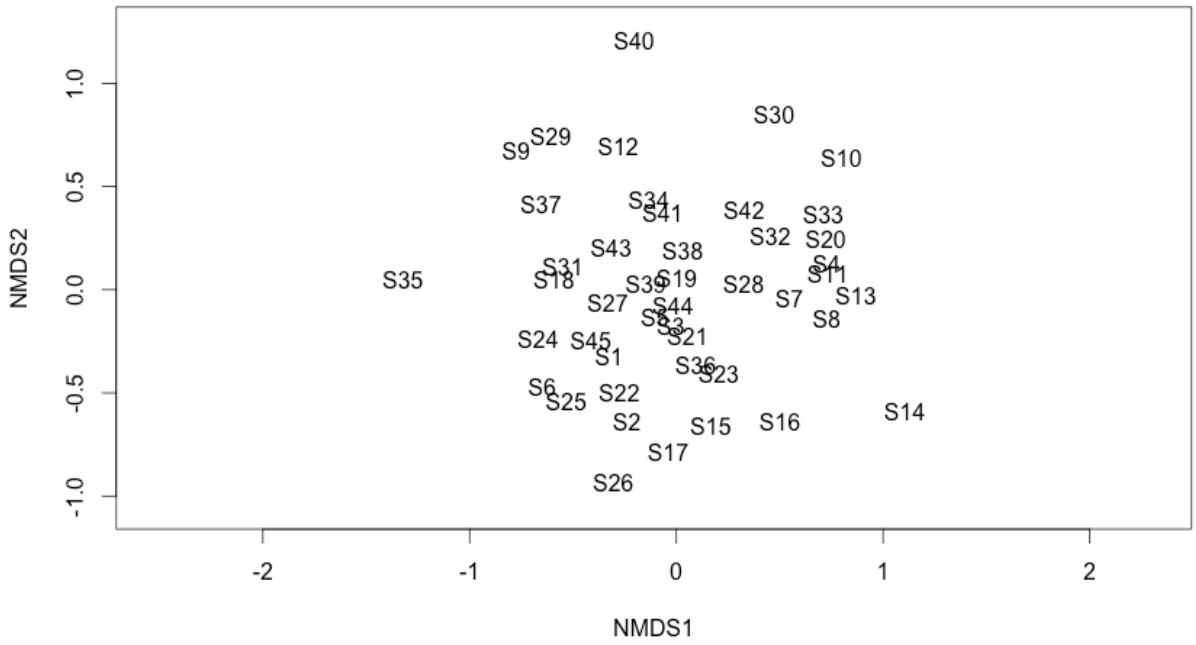
How good a job does NMDS do? Well, another way to determine this is to look at the correlation between the calculated dissimilarities and the plotted values (after all, that's what it's trying to maximize). Specifically, we can plot the relationship between original dissimilarities and Euclidean distances in the ordination using the stressplot() function. Try typing:

```
stressplot(spe.nmds2)
```



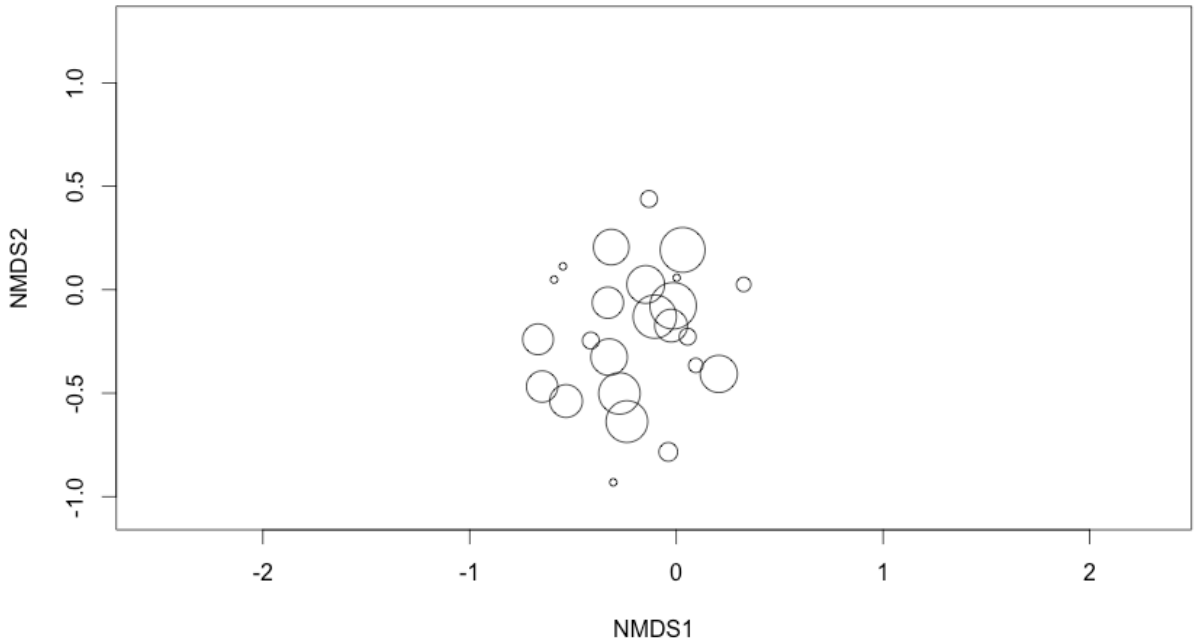
Finally, let's examine the 2-dimensional NMDS configuration for presentation purposes. First we will plot the objects (sites) in ordinate space to visualize the default settings.

```
plot(spe.nmds,type='n')
text(spe.nmds,labels=row.names(speabu))
```



Say we'd like to see how a particular descriptor (in this case, rosyside dace abundance) changes with location. We can make the symbol size proportional to log abundance. Try typing,

```
plot(spe.nmds,type='n')
points(spe.nmds,cex=speabu.log$ROSYDACE)
```



Notice that sites without rosyside dace (i.e., abundance = 0) are not depicted on the plot because their symbol size is zero. The range of options for depicting ordination figures is enormous and beyond the scope of this document. Other ordination graphing functions are available in the vegan package (e.g., **ordiplot**) and include many useful examples that may help you present your own data.

NMDS produces sample scores which are the coordinates of the samples in the k-dimensional ordination space, and these are stored in the result object list in the component named 'points'. To see the sample scores, type:

```
spe.nmds$points
```

Calculate the loadings (i.e., variable weights) on each NMDS axis

To calculate and depict species loadings (i.e., variable weights) on each derived axis from the NMDS we'll use the function envfit() along with the NMDS scores (recall we used the same function for PCoA). The function envfit() simply performs a linear correlation analysis based on standardized data (in other words, a simple linear regression) between each of the original descriptors (i.e., species) and the scores from each NMDS axis. A permutation test is used to assess statistical significance, rather than using the F distribution.

```
vec.sp<-envfit(spe.nmds$points, speabu.log, perm=1000)
```

This should return the values (note yours will be slightly different because it is based on random permutations) listed below for the first 8 species:

```
vec.sp
```

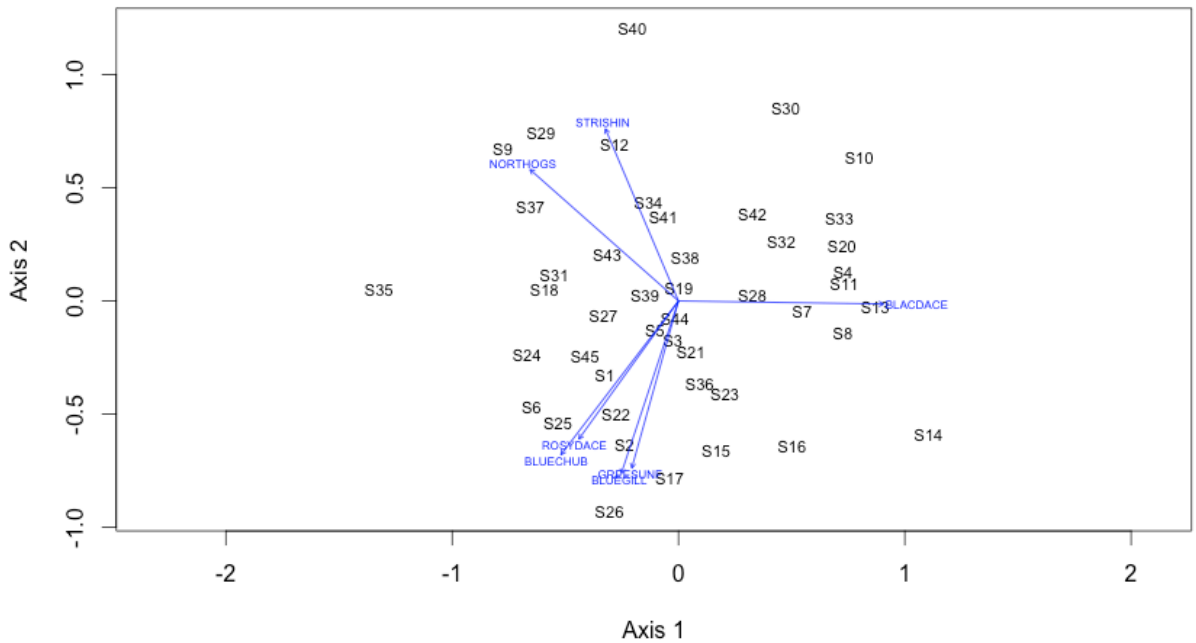
```
***VECTORS

              Dim1      Dim2     r2 Pr(>r)
BANDDART -0.521765  0.853089 0.2247  0.003 **
BANDSCUL -0.278855  0.960333 0.0803  0.175
BLACDACE  0.996128 -0.087917 0.4085 <0.001 ***
BLUECHUB -0.651379 -0.758753 0.3852 <0.001 ***
BLUEGILL -0.380731 -0.924686 0.3195 <0.001 ***
BLUNMINN -0.999536  0.030458 0.0455  0.384
BROOTROU  0.942122 -0.335271 0.0646  0.230
CCHUBSUC -0.123188 -0.992383 0.1760  0.018 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
P values based on 1000 permutations.
```

Based on these results you will notice that banded darter (BANDDART), black dace (BLACDACE), blue chub (BLUECHUB), bluegill (BLUEGILL) and creek chub sucker (CCHUBSUC) and common shiner (COMMSHIN) show statistically significant loadings on the first two principal components. These species could be used to interpret the position of the stream sites (objects) in ordination space.

Now, plot these loadings on the ordination plot.

```
ordiplot(spe.nmds, choices = c(1, 2), type="text", display="sites",
xlab="Axis 1", ylab="Axis 2")
plot(vec.sp, p.max=.01, col="blue")
```

where p.max is the significance level that the species occurrence data must have with either axis in order to be depicted (these p-values were presented in vec.sp).



Combining clustering results and ordination

It is time to start integrating across multivariate approaches. Let's do this by symboling the objects (sites) in the NMDS ordination plot according to clusters idenitied using hierarchical clustering.

First, let's conduct Ward clustering of the species abundance matrix according to Bray-Curtis dissimilarity by typing:

```
speabu.d<-vegdist(speabu.log, "bray")
sitecl.ward<-hclust(speabu.d,method='ward.D2')
sitecl.class<-cutree(sitecl.ward,k=4)
groups<-levels(factor(sitecl.class))
```

Now let's combine with the NMDS results by typing:

```
site.sc <- scores(spe.nmds)
p <- ordiplot(site.sc, type="n", main="NMDS combined with clustering")
for (i in 1:length(groups))
{
  points(site.sc[sitecl.class==i,], pch=(14+i), cex=2, col=i+1)
}
text(site.sc, row.names(speabu), pos=4, cex=0.7)
```

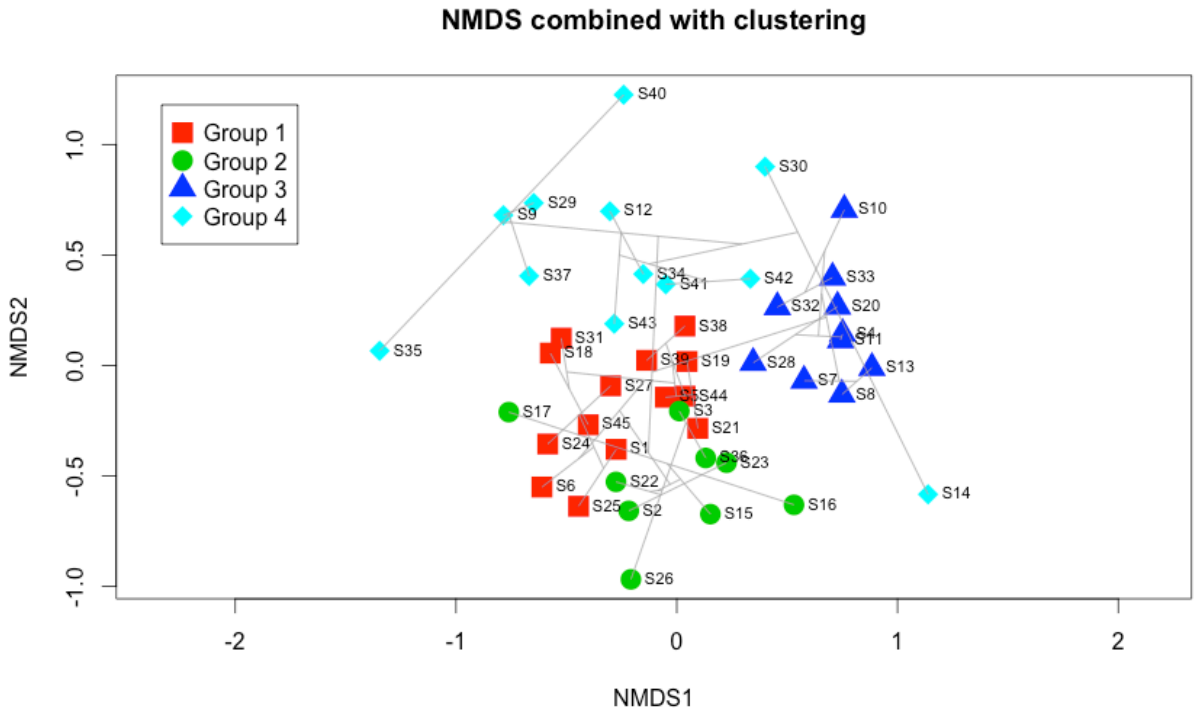
We can add the dendrogram results (optional) by typing:

```
ordicluster(p, sitecl.ward, col="dark grey")
```

Finally, we can a legend interactively by typing:

```
legend(locator(1), paste("Group",c(1:length(groups))),
pch=14+c(1:length(groups)), col=1+c(1:length(groups)), pt.cex=2)
```

Use your mouse to click on the preferred location of the legend.



OPTIONAL READINGS (* recommended)

Kruskal, J.B. 1964a. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29: 1–27.

Kruskal, J.B. 1964b. Nonmetric multidimensional scaling: a numerical method. Psychometrika 29: 115–129.

Kruskal, J. B., and Wish. M. 1977. Multidimensional Scaling. Sage Publications. Beverly Hills. CA.

James, F.C. and McCulloch. 1990. Multivariate analysis in ecology and systematics: Panacea or Pandora's box. Annual Review in Ecology and Systematics 21:129-166.

EXERCISE

Purpose

Upon completion of this chapter, you should be able to do the following: (1) Carry out a NMDS; (2) assess the most powerful solution with respect to the number of dimensions; (3) interpret NMDS axis scores; and (4) consider how NMDS scores may be used in further analyses.

Tasks

- Perform a NMDS using an appropriate resemblance matrix.
    - Is a 2-dimensional solution suitable? What about 3-dimensions?
    - How does the maximum number of random starts affect your ability to find a stable solution?
    - Interpret the bi-plot (i.e., ordination with object scores and variable weights)

9. Principal Component Regression

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Using the scores from a PCA in multiple regression

R Packages: vegan
R Source: biostats

BACKGROUND

Principal component analysis has played an important role in ecology by providing an indispensable analytical tool for elucidating structure in inherently complex datasets. Principal component scores (i.e., the set of orthogonal or independent coordinates derived from the principal component analysis) or scores from PCoA are often visually inspected in reduced-ordination space, or used to summarize the major patterns of variation in the data prior to conducting post hoc statistical analyses (e.g., regression analysis, ANOVA designs: see Jackson 1991; Jolliffe 2002).

Increasingly, ecologists are using the principal components rather than the original explanatory variables in regression analyses; an approach called principal component regression (PCR). In community ecology, for example, fish-habitat relationships have been examined by regressing a biological variable (e.g., species abundance, species richness) against the principal components derived from an environmental dataset (e.g., Kodric-Brown & Brown 1993; Zampella & Bunnell 1998; Marsh-Matthews & Matthews 2000; Romanuk & Kolasa 2002 are just a few examples). For example, Marsh-Matthews & Matthews (2000) used PCR to explore the association between fish species richness and landscape factors (over 30 variables summarized by a few principal components) for streams in midwestern United States. The use of composite scores from a subset of principal components (see Jolliffe 1982; 2002 for a review of principal component selection in regression) has the prospective advantage of alleviating a number of statistical problems associated with regression analysis, including the deleterious effects of multicollinearity (e.g., biased and imprecise estimates of the regression coefficients), biased model selection (e.g., erroneous selection of random variables in the regression model when examining large numbers of variables) and the interaction between the two (e.g., failure to identify significant variables). Therefore, it has been suggested that PCR may provide a simple solution to the negative influence of multicollinearity in regression analysis, and result in models with more stable estimates of variable importance and greater predictive power compared to using the original explanatory variables (Jolliffe 2002).

Here, let's conduct a simple Principal Component Regression that quantifies the relationship between stream habitat characteristics and fish species richness in the class dataset.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**. Import the dataset by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv', header=TRUE, row.names=1)
```

Let's transform the data before diving into the analysis. We will do this because the species abundance dataset is highly skewed and contains some rather large values which are valid but highly influential. Remember, the log of zero is undefined so we'll add 1 to each value in our data set.

PRINCIPAL COMPONENT REGRESSION

First, let's conduct a PCA on the environment dataset following the instructions provided in Chapter 6.

```
env.pca<-prcomp(envdata, scale=TRUE)
summary(env.pca)
```

This returns the following information:

```
Importance of components:
                          PC1   PC2   PC3   PC4    PC5    PC6    PC7    PC8
Standard deviation     1.655 1.377 1.203 1.168 0.8916 0.7472 0.7225 0.5870
Proportion of Variance 0.274 0.190 0.145 0.136 0.0795 0.0558 0.0522 0.0345
Cumulative Proportion  0.274 0.463 0.608 0.745 0.8241 0.8799 0.9321 0.9666
```

Note that the first 4 PCs explain almost 75% of the variation in the original environmental dataset, therefore we will use the principal component scores from these axes to represent the dominant gradients of environmental variation. Recall from Chapter 6 that we can extract these scores by simply typing:

```
env.scores<-env.pca$x
```

Next, let's conduct a regression analysis of fish species richness against the composite PCs representing the majority of the variation in the original environmental dataset.

Calculate species richness for each site:

```
sperich<- rowSums(speocc)
```

Merge the species richness vector with the PC scores:

```
data<-data.frame(cbind(sperich,env.scores))
```

Conduct a simple linear regression relating the first four PCs to species richness:

```
reg<-lm(sperich~PC1+PC2+PC3+PC4,data)
summary.lm(reg)
```

This will result in the following regression output

```
Call:
lm(formula = sperich ~ PC1 + PC2 + PC3 + PC4, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3941 -2.3419 -0.1691  1.9075  8.5136

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.7333     0.5232  16.693  < 2e-16 ***
PC1          -0.3869     0.3197  -1.210 0.233248
PC2          -0.6622     0.3843  -1.723 0.092577 .
PC3           0.9513     0.4398   2.163 0.036559 *
```

```
PC4           1.6824     0.4530   3.714 0.000623 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.51 on 40 degrees of freedom
Multiple R-squared: 0.3641,     Adjusted R-squared: 0.3005
F-statistic: 5.726 on 4 and 40 DF,  p-value: 0.0009677
```

OPTIONAL READINGS

Jolliffe, I.T. 1982. A note on the use of principal components in regression. Applied Statistics 31: 300-303.

Jolliffe, I.T. 2002. Principal Component Analysis, 2nd ed. Springer-Verlag, New York.

EXERCISE

Purpose

Upon completion of this chapter, you should be able to complete a Principal Component Regression.

Tasks

- Conduct a Principal Component Regression to analyze the relationship between fish species richness and derived PCs based on stream environmental characteristics.
    - What does the output from the regression tell you regarding the fish richness-habitat relationship?
    - Can you determine which environmental variables are the most important for predicting species richness?
    - Can you better predict species richness using a Principal Component Regression approach or a traditional regression using the raw environmental variables?

10. Correspondence Analysis (CA) and Detrended CA (DCA)

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Perform CA and DCA, run diagnostics, generate biplots

R Packages: vegan, FactoMineR, factoextra,
R Source: biostats, evplot

BACKGROUND

Correspondence analysis (CA) is one of two names given to a method of ordination that was developed in the 1970s and continues to be heavily used by ecologists. Interestingly, CA was independently developed by Hill (1973,1974) in England (under the name reciprocal averaging) and by Benzecri (1969) in France (under the name analyse factorielle des corresponences). Legendre and Legendre (1998) trace its origins back farther than that outside of ecology.

Correspondence analysis aims to compare the correspondence between samples and species from a table of counted data (or any dimensionally homogenous table) and to represent it in reduced ordination space. PCA and CA are actually quite similar, both conceptually and statistically (Legendre and Legendre 1998). Whereas PCA operates in Euclidean data space to repartition the total variance as a series of optimized linear additive components (ordination axes), CA partitions the total contingency chi-square (or inertia) as a series of linear additive components within a chi-square data space (Digby and Kempton 1987). Noticeably, instead of maximizing the amount of variance explained by the ordination (PCA), CA maximizes the correspondence (or inertia) between species scores and sample scores. In other words, CA is essentially an eigenanalysis of a chi-square distance matrix. A chi-square distance matrix is defined by the deviation from expectation. The use of the chi-square metric results in a weighted linear mapping of samples (and species) into a reduced ordination space, where the particular weighting scheme (doubly weighting based on row and column totals) assumes a unimodal (rather than linear) species response to the underlying gradients. For this reason, CA is better able to capture ecologically meaningful gradients in species abundance data sets with many zeros and handles presence/absence data with greater ease.

CA is popular among ecologists for at least three reasons. First, this technique is recommended when species display unimodal (bell shaped or Gaussian) relationships with environmental gradients (ter Braak 1985), as it happens when a species favors specific values of a given environmental variable. Second, the reciprocal averaging algorithm disregards species double absences because the relationships between rows and columns of the table are quantified using the chi-square coefficient that excludes double absences (Legendre and Legendre 1998). Third, in contrast to the other ordination methods, CA simultaneously ordinates objects (e.g., sites) and descriptors (e.g., species) using a weighted averaging approach. That is, species centroids (the hypothetical mode of their distribution in ordination space) can be plotted as well as the locations of the samples. Given these advantages, CA does have one slight drawback (not a big deal!), which we will discuss later in relation to detrended correspondence analysis (DCA).

SET-UP

In this exercise you will be working with the MAHA species presence-absence dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**. There are two ways that we can import the species presence-absence data.

First, we can transform the species abundances into presence/absence (i.e., binary transformation) using the power method with an exponent equal to zero, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
speocc <- data.trans(speabu,method='power',exp=0,plot=F)
```

Alternatively we can directly import the data, by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
```

CONDUCTING CORRESPONDENCE ANALYSIS

We will use the cca() function in the vegan library, which allows for both unconstrained correspondence analysis (CA) as well as the constrained version, canonical correspondence analysis (CCA), which we will address in a subsequent chapter. You can also use the decorana() function. The cca() function accepts matrices or data frames, computing "scores" for each sample and each species in the matrix or data frame. The scores are derived from the corresponding eigenvectors or, equivalently, from the weighted averages of the opposing scores. The cca() function expects the input matrix or data frame to contain only the samples and species of interest. Thus, make sure the data set has just the desired set of samples and species, and make sure that the data have already been transformed, if appropriate. Note, CA has a built in chi-square distance, so a row or column standardization may not be appropriate. The usage of cca() is:

cca(formula, data, ...)
cca(X, Y, Z, ...)

Where:

- **formula** Model formula, where the left hand side gives the community data matrix, right hand side gives the constraining variables, and conditioning variables can be given within a special function Condition.
- **data** Data frame containing the variables on the right hand side of the model formula.
- **X** Community data matrix.
- **Y** Constraining matrix, typically of environmental variables. Can be missing.
- **Z** Conditioning matrix, the effect of which is removed (`partialled out') before next step. Can be missing.
- **Scale** species to unit variance (like correlations).
- **...** Other arguments for print or plot functions (ignored in other functions).

For additional documentation on cca() type **`?cca`**.

Let's perform CA on the species presence-absence matrix, by typing:

```
spe.ca<-cca(speocc)
```

The results of a CA are complex and stored in a list, including:

- eigenvalues (inertia) associated with each axis
- eigenvectors (sample and species scores)

Type **`summary(spe.ca)`** to view the summary of the eigenvalues. Your results should be as below.

```
Call:
cca(X = speocc)
```

```
Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total           2.845          1
Unconstrained   2.845          1

Eigenvalues, and their contribution to the mean squared contingency
coefficient

Importance of components:
                         CA1    CA2     CA3     CA4    CA5     CA6
Eigenvalue            0.4013 0.3089 0.26673 0.20120 0.1934 0.17992
Proportion Explained  0.1410 0.1086 0.09375 0.07072 0.0680 0.06324
Cumulative Proportion 0.1410 0.2496 0.34338 0.41409 0.4821 0.54533
…

Scaling 2 for species and site scores
* Species are scaled proportional to eigenvalues
* Sites are unscaled: weighted dispersion equal on all dimensions

Species scores

              CA1      CA2       CA3       CA4      CA5      CA6
BANDDART  2.75402 -0.78122  0.005164 -0.424512  0.38277 -0.44415
BANDSCUL  0.70288  0.16340 -0.222250  0.281777 -0.80426  1.15851
BLACDACE -0.20830  0.44562 -0.214540 -0.342707 -0.31654  0.16223
…

Site scores (weighted averages of species scores)

          CA1      CA2      CA3      CA4      CA5      CA6
S1  -0.398438 -0.13939  1.17466  0.81799 -1.08594 -0.17203
S2  -1.110843 -0.94457  0.13356 -1.17710  1.08426  0.22256
S3  -0.415313  0.02718  0.69385 -0.57723 -0.66535 -0.32217
```

Eigenvalues and Inertia

Recall that in PCA eigenvalues represent 'variance' explained. In CA, eigenvalues represent 'inertia', where total inertia equals the chi-squared statistic of the data matrix standardized to unit total. Thus, the eigenvalues are something akin to variance, but they are not variances exactly. The chi-square statistic is a measure of association between samples and species. If species are not independently distributed among samples, so that species tend to co-occur in samples, the chi-square statistic will be large. The larger the chi-square statistic, the greater the correspondence of species' distributions among samples, and vice versa, the thus the greater the inertia.

To see the inertia (eigenvalue) of each axis, type:

spe.caCAeig

Note the syntax above. The results of the CA were stored in an object called 'spe.ca'. This object is a list containing several components, including one called 'CA'. This component is actually another list containing several components. Type **names(spe.ca)** to see the complete list, including the eigenvalues ('eig'). The call requests the list component named 'eig' in the list component named 'CA' in the object named 'spe.ca'.

The total inertia is equal to the sum of all the eigenvalues, and you can get this by typing either one of the commands below:

```
sum(spe.ca$CA$eig)
spe.ca$CA$tot.chi
```

Recall that you divide each eigenvalue by the sum of eigenvalues to calculate the proportion of variation accounted for by each axis. To calculate this as a percentage you can type:

```
spe.ca$CA$eig/sum(spe.ca$CA$eig)*100
```

This will return the following (just the first 6 axes are shown):

```
       CA1          CA2          CA3          CA4          CA5          CA6
14.10396566 10.85837238  9.37518000  7.07179790  6.79953686  6.32383406
```

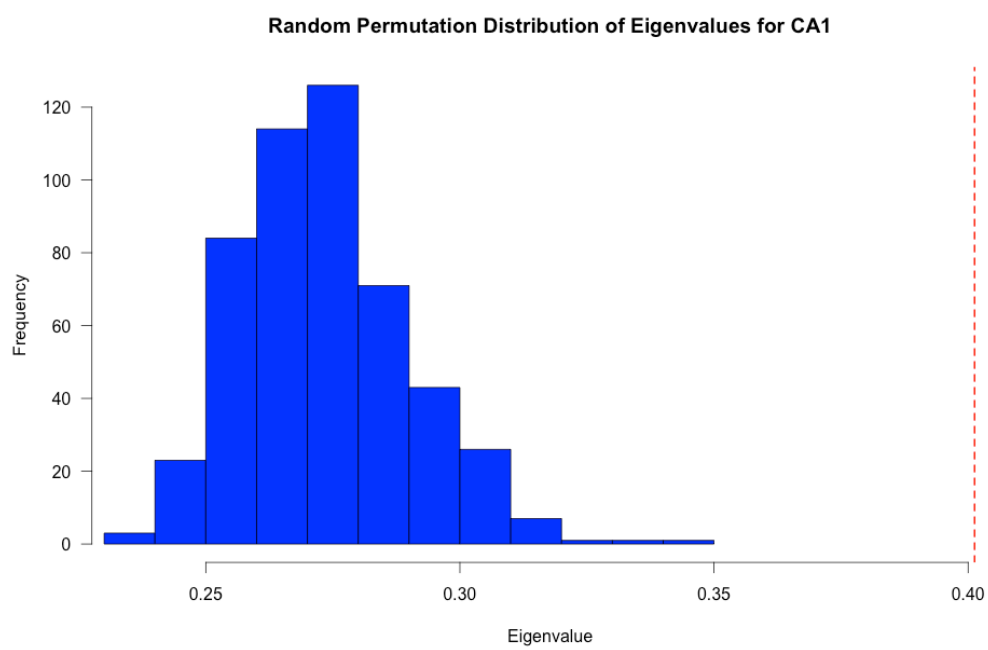
Note that all of this information is also presented in the summary output (see above). I just wanted to make sure you know where the values are coming from!

We may wish to test the statistical significance of the first several eigenvalues using a randomization test, as follows:

```
ordi.monte(speocc,ord='ca',dim=5,perm=500)
```

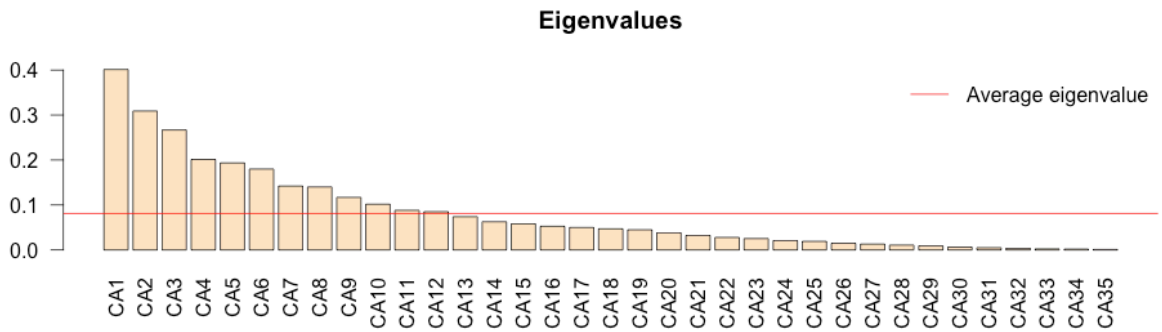
This will return the table below as well as histograms of null and observed eigenvalues for each axis. Axis 1 results are shown.

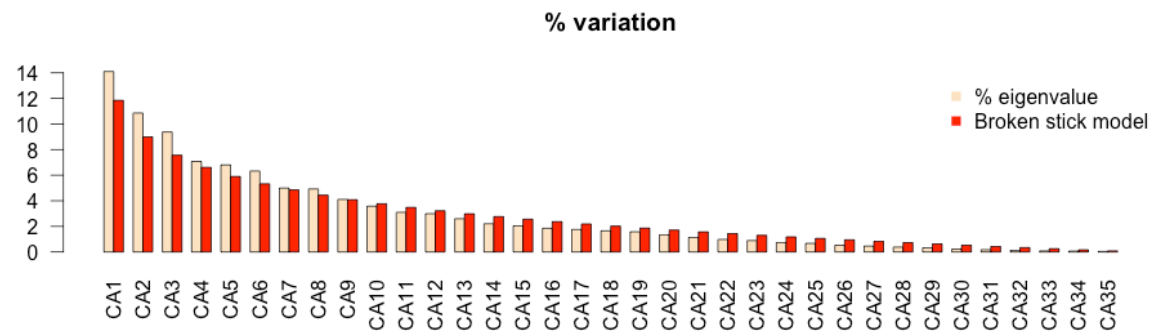
```
Randomization Test of Eigenvalues:
             CA1   CA2   CA3   CA4   CA5
Eigenvalue 0.401 0.309 0.267 0.201 0.193
P-value    0.000 0.000 0.000 0.810 0.442
```



We can look at this same information graphically with the evplot() function (DON'T FORGET TO SOURCE THIS FILE):

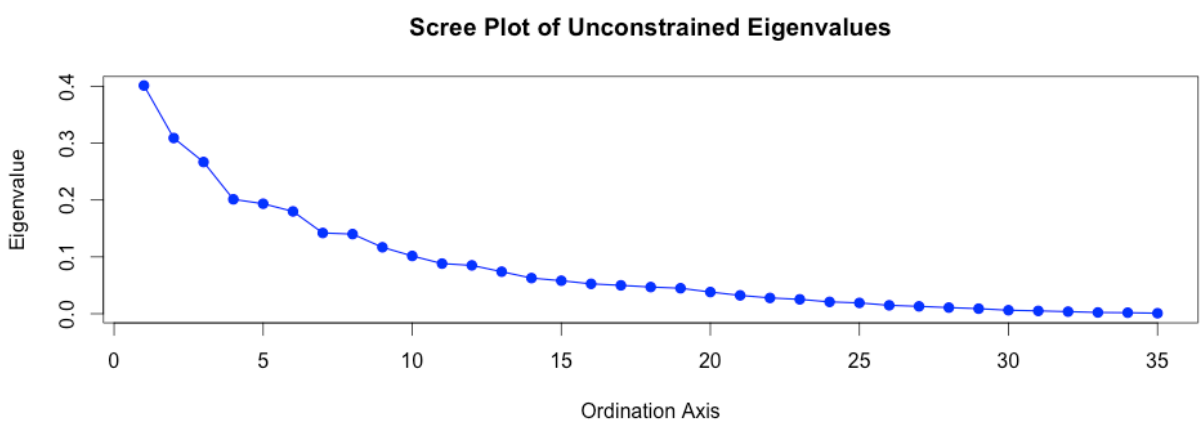
```
evplot(spe.ca$CA$eig)
```





We can look at this same information graphically with the ordi.scree() function:

```
ordi.scree(spe.ca,ord='ca')
```



Site and species scores

To see the sample and species scores for the first two axes, type:

```
spe.ca$CA$u[,1:2]        # sample scores ('u')
spe.ca$CA$v[,1:2]        # species scores ('v')
```

Here are the scores for the first 5 sites:

```
             CA1         CA2
S1  -0.398437934 -0.13939364
S2  -1.110842589 -0.94456616
S3  -0.415313410  0.02718452
S4  -0.711761834  2.08266581
S5  -0.134982565  0.90637210
…
```

Note, in contrast to PCA, the structure correlations (i.e., linear correlations between the species, in this case, and the ordination axes) are not meaningful because we are assuming a unimodal response function. Therefore, there is little point on trying to interpret the axes on the basis of which species are loaded highly. Instead, we are interested in the relative positions of species in the ordination space and, reciprocally, the relative positions of objects in the ordination space.

Ordination plots

Recall that there are lots of interesting options for displaying relationships in ordination plots, including:

- simple plots of samples and species
- displaying related samples (groups) on ordination plots
- fitting and displaying predictor variables on ordination plots

All of the plotting functions used in the PCA lab are applicable here and operate identically. Just substitute the CA object (spe.ca) for the PCA object.

Issues in scaling: CA has several conventions for scaling the sample and species scores. Recall that CA is a weighted averaging method, therefore both samples and species are often jointly depicted in the ordination space (i.e. joint plot), where the center of inertia (centroid) of their scores corresponds to the zero for all axes. Depending on the choice of the scaling type, either the ordination of rows (samples) or the columns (species) is more meaningful, and can be interpreted as an approximation of the chi-square distances between samples or species, respectively.

In scaling type 1, sample scores are calculated as weighted averages of species scores; i.e., the means of the species scores for species that occur across the sites. Sample points that are close to each other in ordination space are similar with regard to the pattern of relative frequencies across species. This scaling is the most appropriate if one is primarily interested in the ordination of objects.

In scaling type 2, species scores are calculated as weighted averages of samples scores; i.e., the means of the sample scores in which the species occurs. Species points that are close to each other in ordination space are similar with regard to the pattern of relative frequencies across sites. This scaling is the most appropriate if one is primarily interested in the ordination of descriptors.

Note, when we take a weighted average, the range of averages shrinks from the original values. The shrinkage factor is equal to the eigenvalue of CA, which has a theoretical maximum of 1. Thus, in scaling type 1, the sample scores are scaled by the eigenvalues, whereas in scaling type 2, the species scores are scaled by the eigenvalues.

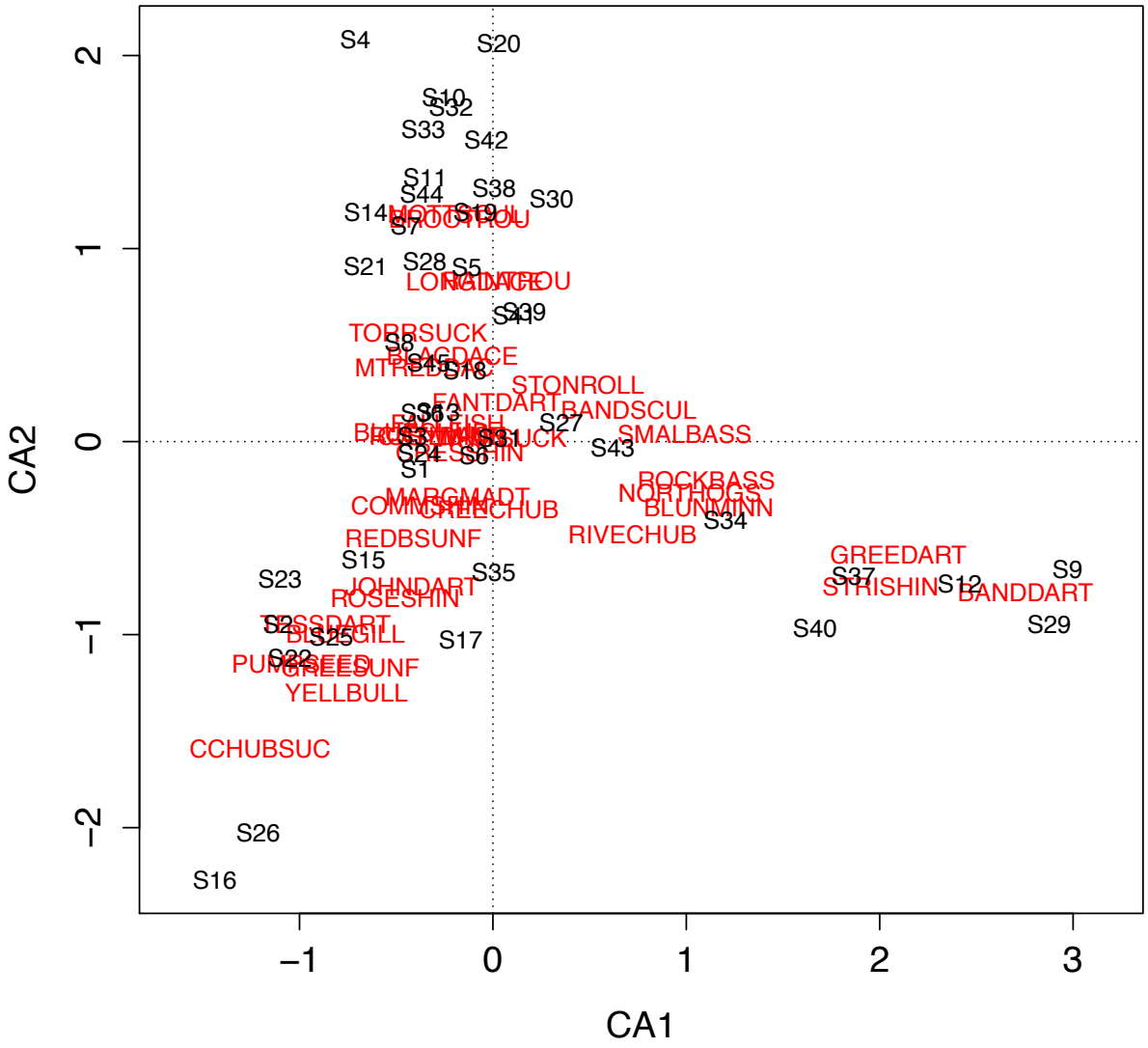
In scaling type 3, the sample and species scores are scaled symmetrically; both are scaled by the square root of eigenvalues.

Here, let's create a CA joint-plot using scaling type 2 by typing:

```
ordiplot(spe.ca,choices=c(1,2),scaling=2)
```

You might be interested in labeling the plot above with object (site) names to help with interpretation.

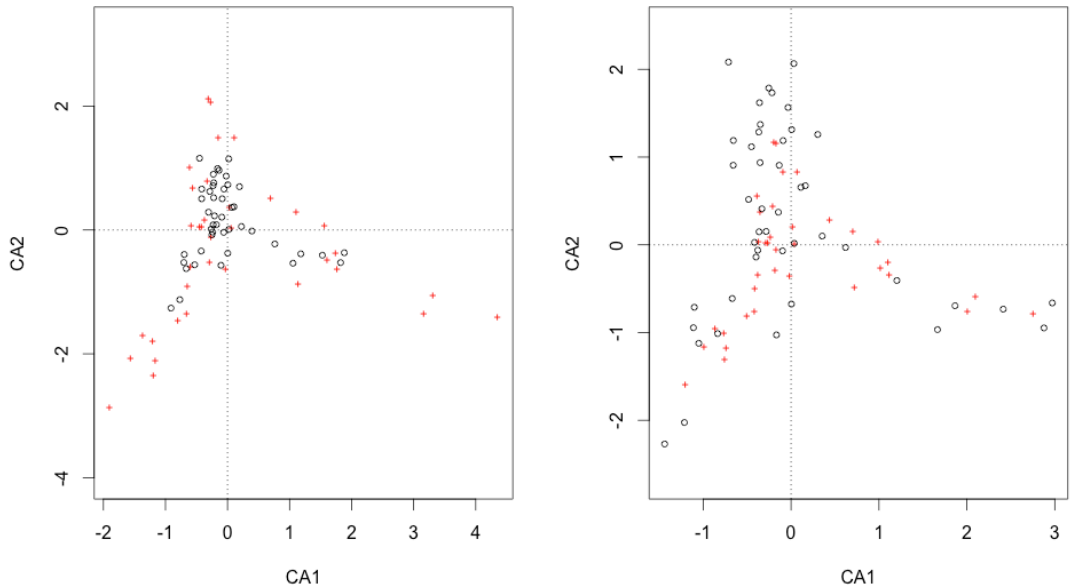
```
ordiplot(spe.ca,type='t', scaling=2)
```



You might also interested in comparing the two scaling approaches. To do this you can type:

```
par(mfrow=c(1,2))
```

```
ordiplot(spe.ca, scaling=1)
ordiplot(spe.ca, scaling=2)
```

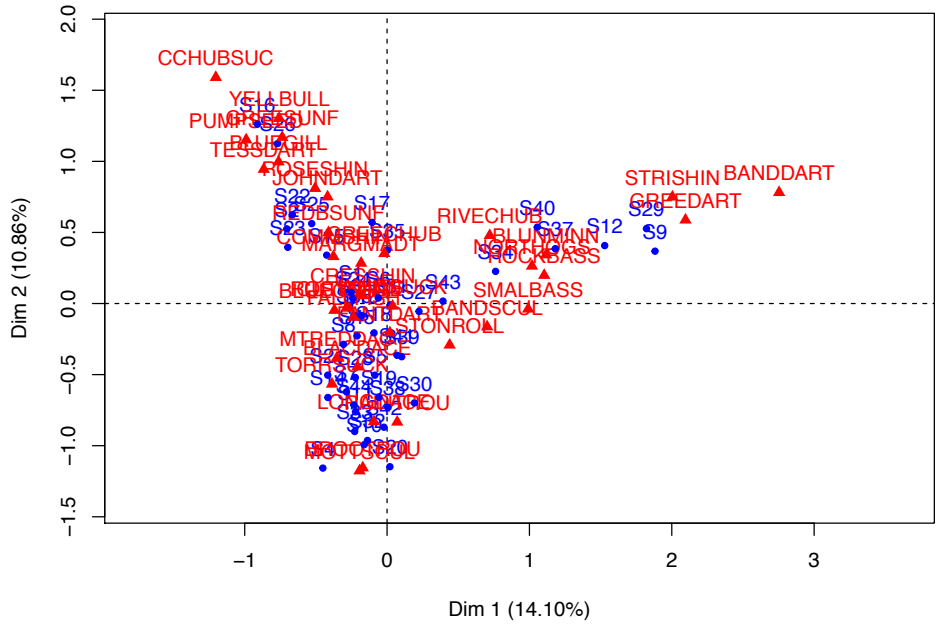


Additional visualization options

Let's use the FactoMineR and factoextra packages to produce some nicer looking outputs from CA. In a nutshell, factoextra is an **R** package to easily extract and visualize the output of multivariate data analyses conducted in FactoMineR.

First type the following to conduct the CA:

```
spe.ca2<-CA(speocc)
```



Type the following to obtain the other results

```
summary(spe.ca2)
```

DETRENDED CORRESPONDENCE ANALYSIS

When the species composition of the sites progressively changes along the environmental gradient, sample positions may appear in the ordination plot as nonlinear configurations called 'arch' (Gauch 1982) (or 'horseshoe' in the case of PCA), which may impair further ecological interpretation. In CA, the arch effect may be mathematically produced as a side-effect of the CA procedure that tries to obtain axes that both maximally separate species and that are uncorrelated to each other (ter Braak 1987): when the first axis suffices to correctly order the sites and species, a second axis (uncorrelated with the former) can be obtained by folding the first axis in the middle and bringing its extremities together, thus resulting in an arch configuration.

To remove the arch effect in CA, a mathematical procedure, detrending, is used to flatten the distribution of the sites along the first CA axis without changing their ordination on that axis. The approach is then designated as detrended correspondence analysis (DCA).

Specifically, DCA attempts to correct each of these drawbacks with the following solutions:

- Single long gradients appear as curves or arcs in ordination: the solution is to detrend the later axes by making their means equal along segments of previous axes.

- Sites are packed more closely at gradient extremes than at the center: the solution is to rescale the axes to equal variances of species scores.

- Rare species seem to have an unduly high influence on the results: the solution is to downweight rare species.

The execution of DCA and the examination of the results are almost identical to those described already for CA, and we will not repeat the procedures again here. Briefly, we will use the function decorana() to conduct DCA. Its usage is:

decorana (veg, iweigh=0, ira=0, mk=26)

Where:

6. **veg** is the community data set
7. **iweigh** indicates whether rare species should be downweighted (0, no; 1, yes). Default is 0.
8. **ira** indicates the type of analysis (0, DCA; 1, CA). Default is 0.
9. **mk** indicates the number of segments in rescaling. Default is 26. This option is only invoked when ira=0.

For additional documentation on decorana type **`?decorana`**.

Now, let's conduct a DCA by typing:

```
spe.dca<-decorana(speocc,ira=0)

spe.dca

Call:
decorana(veg = speocc, ira = 0)

Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                  DCA1   DCA2   DCA3   DCA4
Eigenvalues     0.3674 0.2450 0.2239 0.1656
```

```
Decorana values 0.4013 0.2320 0.1961 0.1197
Axis lengths    3.1419 3.1044 3.7364 2.8201
```

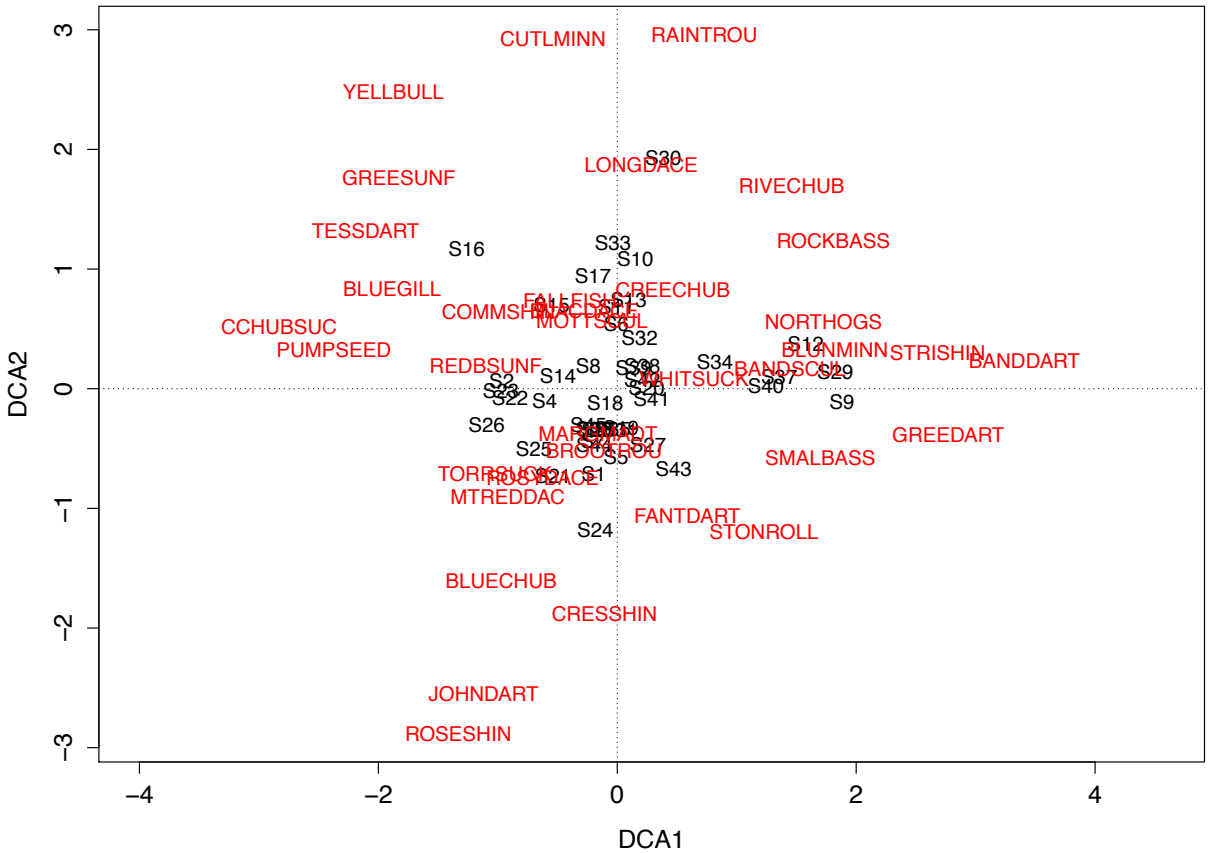
From the above summary we can see that DCA returns eigenvalues and decorana values. Unlike CA, these results are not straightforward to interpret because they do not equate to the proportion of variance explained (because of the detrending and rescaling of axes).

Site and species scores can be obtained by typing:

```
summary(spe.dca)
```

The DCA ordination plot can be obtained by typing:

```
plot(spe.dca)
```



MULTIPLE CORRESPONDENCE ANALYSIS

Multiple correspondence analysis is an extension of the simple correspondence analysis for summarizing and visualizing a data table containing more than two *categorical variables*. We will not explore this approach, but I recommend the FactoMineR library for the analysis and the factoextra library for ggplot2-based elegant visualization.

OPTIONAL READINGS (* recommended)

Benzecri, J.P. 1969. Statistical analysis as a tool to make patterns emerge from data. Pages 35-60 in S. Watanabe [ed.] Methodologies of Pattern Recognition. Academic Press, New York.

Bradfield, G. E. and Kenkel, N. C. 1987. Nonlinear ordination using flexible shortest path adjustment of ecological distances. Ecology 68: 750–753.

Digby, P. G. N. and Kempton. R. A. 1987. Multivariate analysis of ecological communities. Chapman and Hall, London, UK

Greenacre, M. J. 1984. Theory and applications of correspondence analysis. Wiley, New York, NY.

Greenacre, M. J. and Hastie, T. 1987. The geometric interpretation of correspondence analysis. J. Am. Statist. Assoc. 82: 437–447.

Hill, M. O. 1973. Reciprocal averaging: an eigenvector method of ordination. J. Ecol. 61: 237–249.*

Hill, M.O. 1974. Correspondence analysis: a neglected multivariate method. Appl Statis 23: 340–354.

Hill, M.O. and H.G. Gauch. 1980. Detrended Correspondence Analysis: an improved ordination technique. Vegetatio 42: 47-58. *

Jackson, D. A., and K. M. Somers. 1991. Putting things in order: the ups and downs of detrended correspondence analysis. American Naturalist 137: 704-12. *

Legendre, P. and Legendre, L. 1998. Numerical Ecology. Elsevier Science BV, Amsterdam, 853pp.

Palmer, M. W. 1993. Putting things in even better order: the advantages of canonical correspondence analysis. Ecology 74: 2215-30.

Peet, R. K., R. G. Knox, J. S. Case, and R. B. Allen. 1988. Putting things in order: the advantages of detrended correspondence analysis. American Naturalist 131: 924-34.

ter Braak, C.J.F. 1985. Correspondence analysis of incidence and abundance data: properties in terms of a unimodal response model. Biometrics 41: 859–873.

Wartenberg, D., S. Ferson, and F. J. Rohlf. 1987. Putting things in order: a critique of detrended correspondence analysis. American Naturalist 129: 434-48.

EXERCISE

Purpose

Upon completion of this chapter, you should be able to do the following:

(1) Carry out a CA and DCA
(2) Assess the most powerful solution with respect to the number of dimensions; and
(3) Interpret sample and species scores.

Tasks

Perform a CA using the fish species abundance dataset, and address the following:

- How much variation is explained by the first two components? Are these statistical significant?
- Interpret the joint plot (i.e., ordination with object and species scores)
- Compare joint plots using different scaling? How different are they? Does it affect your interpretation of similarities between sites and between species?
- Is there any evidence for arching in the ordination? If so, does a DCA remedy this problem?

11. Canonical Correspondence Analysis (CCA) and Redundancy Analysis (RDA)

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Canonical Correspondence Analysis (CCA), Redundancy Analysis (RDA)

R Packages: vegan
R Source: biostats

BACKGROUND

Canonical methods are used to determine the common structure, or correspondence, between two sets of variables (matrices) measured on the same sampling units (Legendre and Legendre 1998). A typical application involves determining the degree and nature of the relationship between species and environmental variables or the examination of the relationship between two species data sets (e.g., plants and insects). However, the possibilities are endless. In all cases, the objective is to examine and summarize the common structure of the two data sets (Dray et al. 2003). In constrained (canonical) ordination analyses only the variation in the Y matrix (e.g., object-by-species) that can be explained by the X matrix (e.g., object-by-environment) is displayed and analyzed, and not all the variation in the Y matrix. In other words, constrained ordination provides a method of explicitly testing whether an explanatory matrix X significantly explains variation in a response matrix Y. The method is best conceptualized as an extension of linear regression in which our Y matrix corresponds to our response variable and our predictor variables are contained in matrix X. Linear combinations of the explanatory variables are generated in a manner that explains as much variance as possible in the response matrix.

There are three primary approaches to constrained ordination: redundancy analysis (RDA), canonical correspondence analysis (CCA) and constrained analysis on principal coordinates (CAP). CAP is also referred to as distance-based RDA. The most important difference among techniques, at least conceptually, is whether they are based on linear response model (e.g., species are assumed to vary linearly along the measured environmental gradients) or a unimodal or Gaussian response model (e.g., species are assumed to vary in a non-linear and unimodal fashion over sufficiently long environmental gradients).

Redundancy analysis (RDA) is the canonical form of PCA. The method maximizes predictions for a set of response variables (e.g., species data) given a set of predictive variables (e.g., environmental data). The method is essentially a PCA in which the sampling unit locations in species space are restricted to be linear combinations of the predictor or environmental variables. RDA is the appropriate canonical model when both data sets are linear. Because the method involves multiple regression, the number of predictor variables must be small relative to the number of objects and species: otherwise, the results are very similar to unconstrained PCA of the species data (Dray et al. 2003).

Canonical correspondence analysis (CCA) is the canonical form of CA, and is thus closely related to RDA. Like RDA, the method uses multiple regression to obtain linear combinations of the predictor variables that best explain sampling unit positions in species space (ter Braak and Smilauer 2002). CCA is the appropriate model when species responses are unimodal (response variables as contingency data) and the environmental (predictor) data are linear (ter Braak 1986). As in RDA, the number of predictor variables must be small relative to the number of objects.

Constrained Analysis of Principal Coordinates (also called distance-based RDA; Legendre & Anderson, 1999) is the canonical form of PCoA. CAP can be applied to a matrix of distances between objects, thus like PCoA it can use any resemblance metrix. In a nutshell, the technique first applies a PCoA on the

distance matrix to convert it back to a rectangular table containing rows of objects by columns of PCoA coordinates. Those new, uncorrelated coordinates thus correspond to synthetic 'species' variables that are then related to additional environmental parameters using a classical RDA. The function **capscale()** in the vegan library performs CAP.

The R exercise below demonstrates CCA, but the same routines apply to RDA and CAP. The output includes the total variation in species composition as explained by the environmental axes, the cumulative percentage of variance of the species–environment relationship, and the overall statistical significance of the relationships between the species and environmental tables. Canonical models can be represented by a triplot of objects (dots), species (arrows), and environmental variables (arrows for quantitative variables and dots for each level of qualitative or nominal variables), or by any combinations thereof (i.e. biplots) (ter Braak, 1994). Depending on the scaling chosen, i.e. whether the analysis mainly focuses on intersample relationships or interspecies correlations, the interpretation of the relationships between samples, species, and environmental variables differs.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset and environment dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **setwd**, **library** and **source**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
```

Now let's transform both datasets before diving into the analysis. First, let's conduct a 'row normalization' (i.e., to rescale each row so that the sums equal 1, or in other words to calculate relative species abundances), type the following:

```
speabu.tran<-data.stand(speabu,method='total',margin='row',plot=F)
```

Next, let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

DETERMINE THE APPROPRIATE RESPONSE MODEL

The goal of this step is to determine whether a linear or a unimodal response model is appropriate for the community under investigation. There are three basic options:

1. CAP can handle any dissimilarity measure and performs a linear mapping.
2. RDA is based on Euclidean distances and performs linear mapping.
3. CCA is based on Chi-squared distances and performs weighted linear mapping.

Ultimately, the choice of model depends on a multitude of factors; therefore, there is no easy set of guidelines for choosing the "right" model. However, one recommended approach involves conducting a DCA and evaluating the gradient length of the axes. Recall that gradient length in DCA is given as the length of the ordination axis in terms of standard deviations in species scores (equal to the range in site scores divided by the average species tolerance, where tolerance equals the standard deviation in a species distribution of scores along an axis) and is essentially a measure of how much species turnover there is along the full length of the gradient. There are numerous rules of thumb, but in

general a gradient length > 4 guarantees unimodal model as appropriate, 2-4 is probably unimodal, and <2 is probably linear model. To compute gradient lengths type:

```
decorana(speabu.tran, ira=0)

Call:
decorana(veg = speabu.tran, ira = 0)

Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                  DCA1   DCA2   DCA3   DCA4
Eigenvalues     0.6260 0.5526 0.4036 0.4078
Decorana values 0.7581 0.5328 0.3793 0.2681
Axis lengths    5.5389 3.9554 3.3509 3.3394
```

You will notice the axis length for DCA1 is 5.54 standard deviations, which suggest that a unimodal model is the most appropriate. Therefore, throughout this exercise we will focus principally on CCA. However, the procedures given here are virtually identical for RDA. Again, we will employ the familiar functions in the vegan library. In the vegan library, the functions for conducting constrained ordination (RDA, CAPSCALE, CCA) are closely allied. Actually, the results are similarly structured, and they inherit properties from each other. Remember that in deciding whether to use CCA or RDA or CAP to analyze a data set, you should use the same considerations for selecting CA versus PCA versus PCoA (see lecture notes on PCA and CA, Legendre and Legendre 1998 for more details).

CONDUCTING CANONICAL CORRESPONDENCE ANALYSIS

Prior to conducting CCA, it is essential to address a couple data issues that we discussed in class. First, rare species (< 5% of occurrences) should not be included in the analysis given their large influence on the final results. You will have to examine the influence of rare species on your results if you are interested in relaxing this rule-of-thumb (i.e., removing species with < 3% of occurrences). Second, consider all appropriate transformations for both Y (species) and X (environment) matrices. Third, eliminate highly redundant explanatory variables in the X matrix.

In this lab we will use the cca() function in the vegan library, which allows for both unconstrained correspondence analysis (CA) as well as the constrained version, canonical correspondence analysis (CCA). Recall that we used this function in the previous chapter. It's usage is:

cca (X, Y, Z,…)

Where:

10. **X** is the community data set
11. **Y** is the constraining or explanatory matrix
12. **Z** is a conditioning matrix (explained in more detail under "Partial CCA and RDA")

For additional documentation on **`cca`** type **`?cca`**.

Let's perform CCA.

```
spe.cca<-cca(speabu.tran~.,envdata.tran)

summary(spe.cca)
```

```
Call:
cca(formula = speabu.tran ~ Sinuosity + Slope + WDRatio + SubEmbed +
HabQual + Elev + RoadDen + Agricult + HumanUse + BasinAre, data =
envdata.tran)

Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total           6.346     1.0000
Constrained     1.997     0.3146
Unconstrained   4.350     0.6854

Eigenvalues, and their contribution to the mean squared contingency
coefficient

Importance of components:
                         CCA1    CCA2    CCA3    CCA4    CCA5   CCA6
Eigenvalue            0.47804 0.41580 0.31970 0.27330 0.18492 0.1275
Proportion Explained  0.07533 0.06552 0.05037 0.04306 0.02914 0.0201
Cumulative Proportion 0.07533 0.14084 0.19122 0.23428 0.26342 0.2835
…

Accumulated constrained eigenvalues
Importance of components:
                        CCA1   CCA2   CCA3   CCA4    CCA5    CCA6
Eigenvalue            0.4780 0.4158 0.3197 0.2733 0.18492 0.12754
Proportion Explained  0.2394 0.2082 0.1601 0.1369 0.09261 0.06387
Cumulative Proportion 0.2394 0.4476 0.6077 0.7446 0.83721 0.90109
…

Scaling 2 for species and site scores
* Species are scaled proportional to eigenvalues
* Sites are unscaled: weighted dispersion equal on all dimensions
```

Note, the period on the right-hand side of the ~ indicates "all" variables in the specified data set. Alternatively, we could have listed each of the variables on the right-hand side of the equation. The results are equivalent.

Total inertia, eigenvalues and proportion of variation explained

Inertia in CCA corresponds to variance. The proportion of the total inertia captured by the constraining environmental matrix indicates the percent variance explained. Above, 31.5% of the variance is explained by the environmental matrix, i.e., (1.99/6.35)*100. To examine the specific proportion of variance explained by each ordination axis, examine the cumulative variance (accounted) explained by CCA1, CCA2, etc. In this output, lambda refers the eigenvalue, or in other words, the amount of variation. For instance, the first CCA axis explains 7.5% of the variation in the original fish community matrix, and the first and second CCA axis together explains 14.1% of the variation. The cumulative variance explained by each CA axis is indicated on the lower output tier.

Significance tests

It is often useful to test the significance of the ordination. This can be done using Monte Carlo global permutation tests of significance of the canonical axes. Several tests are available.

First, we are usually interested in testing the significance of all constraints simultaneously; that is, whether the proportion of species variance explained by all environmental constraints (i.e., all constrained axes) is significantly greater than expected by chance. The global test can be computed by typing:

anova(spe.cca)

```
Permutation test for cca under direct model

Model: cca(formula = speabu.tran ~ Sinuosity + Slope + WDRatio + SubEmbed +
HabQual + Elev + RoadDen + Agricult + HumanUse + BasinAre, data =
envdata.tran)
         Df  Chisq      F N.Perm Pr(>F)
Model    10 1.9968 1.5609    199  0.005 **
Residual 34 4.3495
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Second, we may also be interested in testing each axis in turn. The test based on the 1st canonical axis has maximum power against the alternative hypothesis that there is a single dominating gradient that determines the relation between the species and environment. This is a sequential test, so that the test of the 2nd axis will partial out the first axis and test the residual variance. Likewise, the test of the 3rd axis will partial out the first two axes and test the residual variance. And so on. Setting the by='axis' argument in the call will perform the tests, as follows:

anova(spe.cca,by='axis')

```
Permutation test for cca under direct model

Model: cca(formula = speabu.tran ~ Sinuosity + Slope + WDRatio + SubEmbed +
HabQual + Elev + RoadDen + Agricult + HumanUse + BasinAre,      data =
envdata.tran)
         Df  Chisq      F N.Perm Pr(>F)
CCA1      1 0.4780 3.7368    199  0.005 **
CCA2      1 0.4158 3.2503    199  0.005 **
CCA3      1 0.3197 2.4990    199  0.005 **
CCA4      1 0.2733 2.1364    199  0.015 *
CCA5      1 0.1849 1.4455     99  0.130
CCA6      1 0.1275 0.9970     99  0.380
CCA7      1 0.0828 0.6472     99  0.880
CCA8      1 0.0625 0.4882     99  0.990
CCA9      1 0.0298 0.2328     99  1.000
CCA10     1 0.0225 0.1757     99  1.000
Residual 34 4.3495
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lastly, we may also be interested in testing each term in the model. Setting by = 'terms' argument will perform a separate significance test for each term (constraining variable). The terms are assessed sequentially from first to last like above, and therefore the order of the terms will influence their significance. Hence, caution must be exercised in using this test unless there is a meaningful order to the terms in the model. The test can be computed by typing:

anova(spe.cca,by='terms')

```
Permutation test for cca under direct model
Terms added sequentially (first to last)
```

```
Model: cca(formula = speabu.tran ~ Sinuosity + Slope + WDRatio + SubEmbed +
HabQual + Elev + RoadDen + Agricult + HumanUse + BasinAre, data =
envdata.tran)
          Df  Chisq      F N.Perm Pr(>F)
Sinuosity  1 0.1363 1.0655     99   0.27
Slope      1 0.3789 2.9618     99   0.01 **
WDRatio    1 0.2661 2.0803     99   0.01 **
SubEmbed   1 0.1823 1.4249     99   0.10 .
HabQual    1 0.1465 1.1449     99   0.31
Elev       1 0.2684 2.0982     99   0.01 **
RoadDen    1 0.1232 0.9628     99   0.51
Agricult   1 0.1192 0.9318     99   0.60
HumanUse   1 0.1107 0.8656     99   0.67
BasinAre   1 0.2652 2.0732     99   0.01 **
Residual  34 4.3495
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ordination scores

Four different sets of ordination scores (coordinates on ordination axes) are produced by the ordination, which are used to portray the relationships among objects, species and explanatory variables in the triplot (below):

Linear combination (LC) sample scores, computed as the predictions from the linear combinations of the independent variables. LC scores are object scores in environmental space, each axis formed by linear combinations of environmental variables.

Weighted averages (WA) sample scores (actually weighted sums in RDA), computed as the weighted averages (sums) of species scores that are as similar to LC scores as possible. WA scores are sample scores in species space.

Species (SP) scores, computed as the weighted average (sum) of sample scores.

Biplot (BP) scores for the explanatory variables (see below).

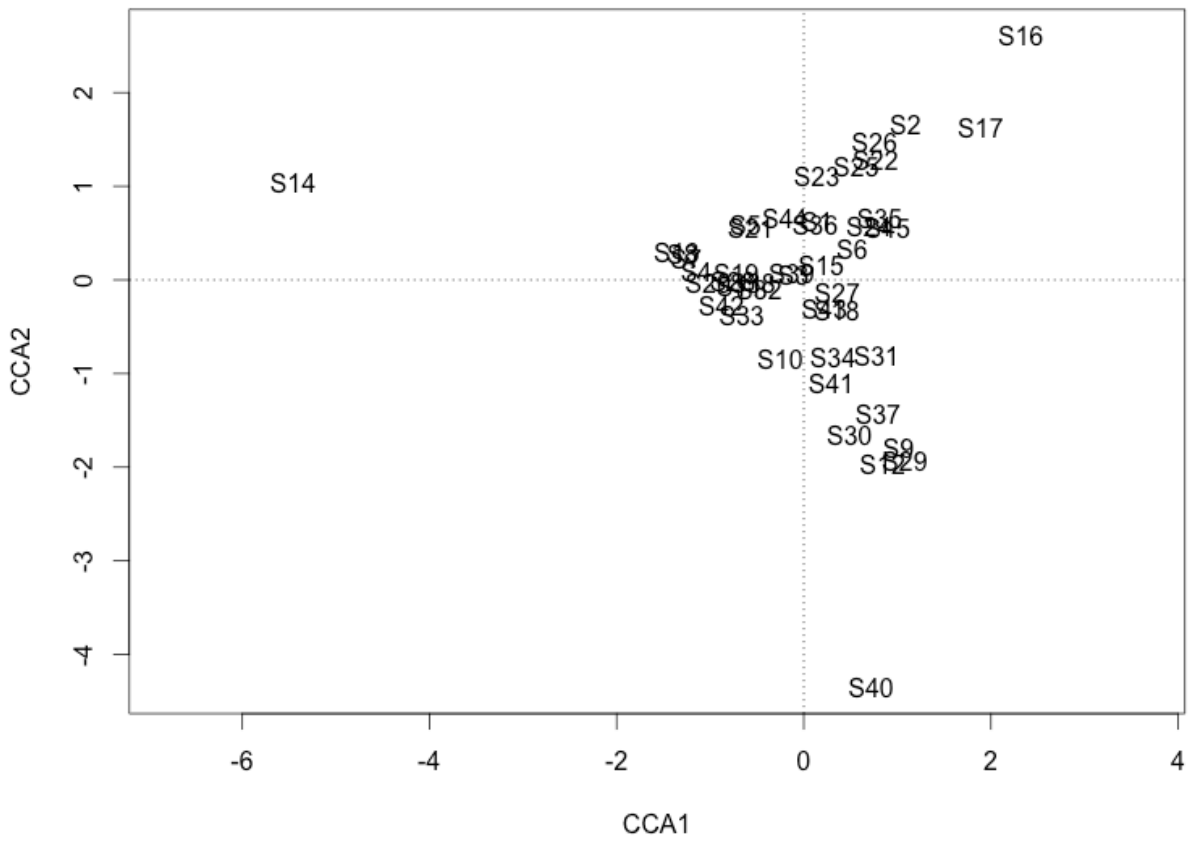
There is a longstanding debate over whether to use LC or WA scores for sites. It has been shown that LC scores are very sensitive to noise in the environmental data, whereas the WA scores are much more stable, being largely determined by the species data rather than being a direct projection in environmental space. Therefore, we will use the WA scores in all subsequent ordinations.

Now, let's visualize the ordination by plotting just the site ('wa') scores.

```
plot(spe.cca,choices=c(1,2),type='points',display='wa',scaling=2)
```

Alternatively, you can just plot the site labels by typing:

```
plot(spe.cca,choices=c(1,2),type='n',display='wa',scaling=2)
text(spe.cca,choices=c(1,2),labels=row.names(speabu))
```



INTERPRETING THE CONSTRAINED ORDINATION

Canonical coefficients

These are the regression coefficients of the linear combinations of environmental variables that define the ordination axes; they are eigenvector coefficients. They are not very useful for interpretation. Nevertheless, they are the coefficients that define the weights of the variables in the linear combinations used to constrain the ordination, and thus ultimately reflect the importance of the variables. However, they are affected by the correlation structure of the explanatory variables and thus do not provide a measure of independent importance of each variable. Note that these are the coefficients of the linear environmental constraints used to generate the sample LC scores.

Intra-set Correlation (structure) Coefficients

Intra-set correlations are between object scores derived as linear combinations of environmental variables (LC scores) and environmental variables (i.e., the variable loadings). Intra-set correlations are related to the canonical coefficients but differ in several important aspects. Both canonical coefficients and intra-set correlations relate to the rate of change in community composition per unit change in the corresponding environmental variable, but for canonical coefficients it is assumed that other environmental variables are being held constant, while for intra-set correlations other environmental variables are assumed to covary as they do in the data set. When variables are inter-

correlated, canonical coefficients become unstable (i.e., arising because of multicollinearity issues); however, intra-set correlations are not affected. You can calculate the intra-set correlations by typing:

```
round(intrasetcor(spe.cca),5)
```

```
              CCA1     CCA2     CCA3     CCA4     CCA5
Sinuosity  0.33292  0.01773  0.09744  0.11041  0.04725
Slope     -0.82642 -0.07313  0.03595  0.42366 -0.16549
WDRatio   -0.00938 -0.64104  0.44340 -0.03867  0.01662
SubEmbed   0.41524 -0.00218  0.29835 -0.14625 -0.48294
HabQual   -0.37318  0.01468  0.42747  0.13428  0.34820
Elev      -0.40426 -0.72487 -0.03802  0.13589  0.49211
RoadDen    0.25732  0.45298 -0.49116  0.07101  0.11144
Agricult   0.30655  0.18215  0.03154 -0.35292  0.31444
HumanUse   0.41061  0.25543 -0.20609 -0.16501  0.13743
BasinAre   0.64416 -0.28667  0.45463  0.36648  0.31819
```

Inter-set Correlation (structure) Coefficients

Inter-set correlations are between species-derived sample scores (WA scores) and environmental variables. Like intra-set correlations, inter-set correlations do not become unstable when the environmental variables are strongly correlated with each other (multicollinear). We can compute them by typing:

```
round(intersetcor(spe.cca),5)
```

```
              CCA1     CCA2     CCA3     CCA4     CCA5
Sinuosity  0.28672  0.01519  0.07705  0.08388  0.03041
Slope     -0.71173 -0.06262  0.02843  0.32186 -0.10653
WDRatio   -0.00808 -0.54894  0.35062 -0.02938  0.01070
SubEmbed   0.35761 -0.00186  0.23593 -0.11111 -0.31087
HabQual   -0.32139  0.01257  0.33803  0.10201  0.22414
Elev      -0.34816 -0.62073 -0.03007  0.10324  0.31678
RoadDen    0.22161  0.38790 -0.38839  0.05395  0.07173
Agricult   0.26400  0.15598  0.02494 -0.26811  0.20241
HumanUse   0.35363  0.21873 -0.16297 -0.12535  0.08847
BasinAre   0.55476 -0.24549  0.35950  0.27841  0.20482
```

Biplot scores for explanatory variables

Environmental variables are typically represented as vectors radiating from the centroid of the ordination. Biplot scores give the coordinates of the heads of the environmental vectors. The environmental vector depicts the direction and magnitude of change in the environmental variable through the ordination space (see below). The biplot scores for the constraints are given in the summary() of the ordination, as follows:

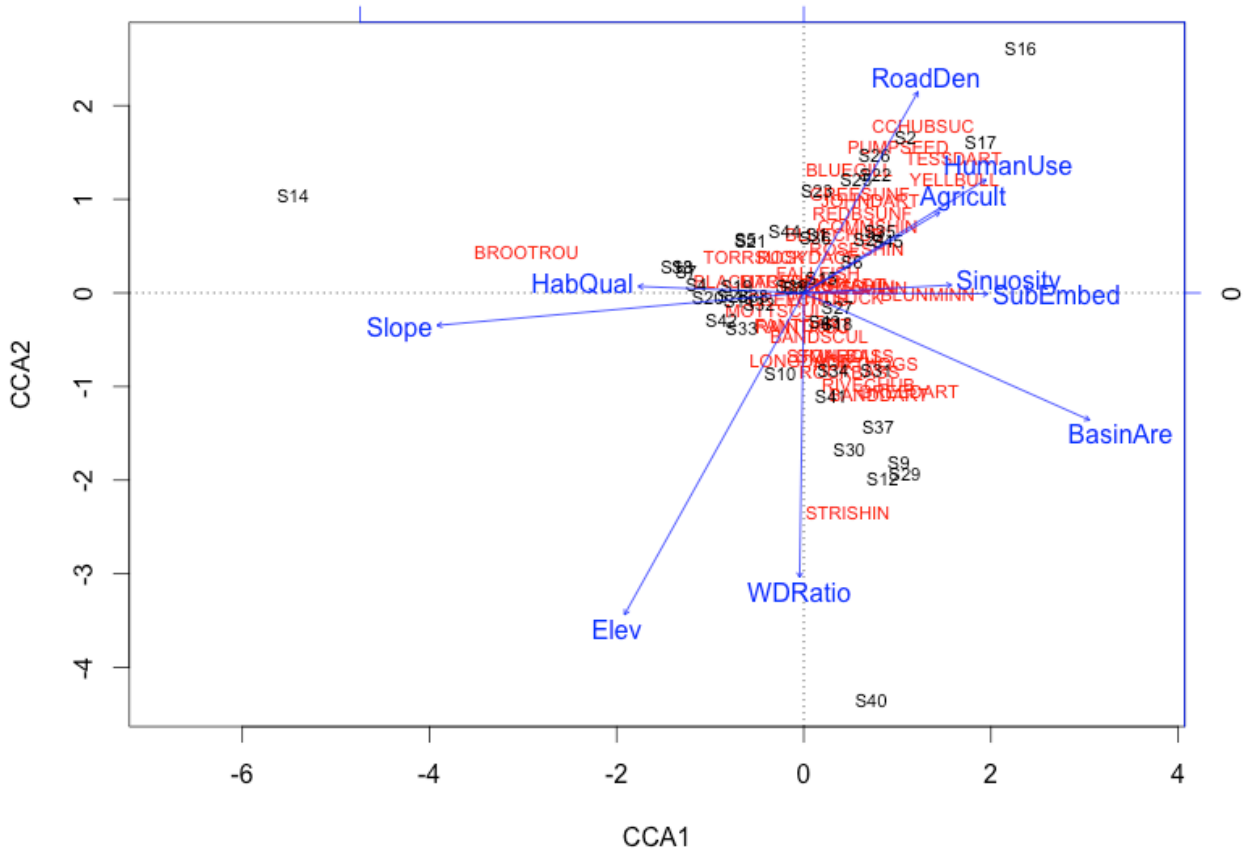
```
summary(spe.cca)
```

Triplot

The triplot displays the major patterns in the species data with respect to the environmental variables. The triplot is the key feature of a constrained ordination and the major means of interpreting the

results. The triplot simultaneously displays the samples (sites), species and environmental constraints. Let's produce a triplot, by typing:

```
plot(spe.cca,choices=c(1,2),display=c('wa','sp','bp'),scaling=2)
```



Note that 'wa' refers to the weighted average site scores, 'sp' refers to the species scores and 'bp' refers to the environmental variable biplot scores. Also note that the arguments in the above call are all the default settings, and therefore we could have created the same triplot by simply typing **`plot(spe.cca)`**, but the above call makes explicit several important things, including the choice of axes, the sets of scores and the scaling of sample and species scores.

As with CA, CCA simultaneously ordinates objects (e.g., sites) and variables (e.g., species). Proximity in ordination space corresponds to similarity in community composition (sites) or patterns of site occupancy (species). The loadings of individual environmental variables are depicted by the blue arrows, where arrow length and direction correspond to the magnitude and gradient of each respective variable. For example, the variable slope (Slope) is negatively related to CCA1, therefore sites on the left side of the ordination figure are characterized by high slopes AND high abundance of brook trout (BROOTROU).

Different scaling of scores

Importantly, there are a number of ways to scale the triplot that affect the exact quantitative interpretation of the triplot. For example, there is no way of scaling species and site scores to each other, so one must choose one or the other to optimize, or attempt to optimize both simultaneously

(symmetric scaling). Each function and ordination program adopt different conventions for scaling species and site scores. The specific differences need not concern us as they merely reflect subtly different ways of expressing the same information. Briefly,

Sample scaling – Optimally displays inter-sample relationships. The variance of sample scores on each axis reflects the importance of the axis as measured by the eigenvalue, whereas the variance of the species scores along the axes are equal. [scaling = 1]

Species scaling – Optimally displays inter-species relationships. The variance of species scores on each axis reflects the importance of the axis as measured by the eigenvalue, whereas the variance of the sample scores along the axes are equal. Since we are typically interested in portraying species relationships, this is generally the preferred scaling and the default in most programs. [scaling = 2]

Symmetric scaling – Achieves a balance between sample and species scaling. In symmetric scaling, both sample and species scores are scaled symmetrically by square root of eigenvalues. [scaling = 3]

Having fun with triplots

There are many options for plotting the results from constrained ordination. In some cases you will have many observations or descriptors and it will not be possible to interpret the default triplot presented above. If you require greater control over the plotting then consider customizing! For example, try typing:

```
plot(spe.cca,choices=c(1,2),type='none',scaling=2)
points(spe.cca,choices=c(1,2),display='wa',pch=19,cex=1.5,scaling=2)
text(spe.cca,choices=c(1,2),display='sp',col='red',cex=.75,scaling=2)
text(spe.cca,choices=c(1,2),display='bp',col='blue')
```

GUIDELINES FOR INTERPRETING ORDINATION TRIPLOTS

Fortunately, regardless of the method (unimodal or linear) and scaling options, there are some common guidelines for interpreting all triplots. These generalized relationships hold up under any of the scenarios above. More importantly, since we are almost always interested in interpreting triplots loosely; that is, interpreting the general relationships among species and between species and environment, the scaling may not actually matter that much because while it changes the display spread of points in the display, the relationships pretty much stay the same.

Here are some general rules for interpreting triplots (modified from McGarigal et al. 2002):

- Sample locations indicate their compositional similarity to each other. Samples tend to be dominated by the species that are located near them in ordination space.
- Species locations indicate their distributional similarity to each other. Species tend to be most present and abundant in the sites that are located near them in ordination space.
- Length of an environmental vector indicates its importance to the ordination. Direction of the vector indicates its correlation with each of the axes. Angles between vectors indicate the correlation between the environmental variables themselves.
- Perpendiculars drawn from species to arrows gives approximate ranking of species response to that environmental variable, and whether a species has a higher-than-average or lower-than-average optimum on that environmental variable.
- Perpendiculars drawn from samples to arrows gives approximate ranking of sample values for that environmental variable, and whether a sample has a higher-than-average or lower-than-average value on that environmental variable
- It is more natural to represent nominal environmental variables by points instead of arrows. The points are located at the centroid of sites that belong to that class. Classes containing sites with high values for a species will tend to lie close to that species point.

OPTIONAL READINGS

Dray, S., Chessel, D. and Thioulouse, J. 2003. Co-inertia analysis and the linking of ecological data tables. Ecology 84: 3078–3089.

Legendre, P. and Anderson, M.J. 1999. Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. Ecol Monogr 69: 1–24.

Legendre, P. and Legendre, L. 1998. Numerical Ecology. Elsevier Science BV, Amsterdam, 853pp.

ter Braak, C. J. F. 1986. Canonical correspondence analysis a new eigenvector technique for multivariate direct gradient analysis. Ecology 67: 1167–1179.

Wollenberg, A.L. 1977. Redundancy analysis, an alternative for canonical analysis. Psychometrika 42: 207–219.

12. Partial Canonical Correspondence Analysis (pCCA) and Partial Redundancy Analysis (pRDA)

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- partial Canonical Correspondence Analysis (pCCA), partial Redundancy Analysis (pRDA)

R Packages: vegan
R Source: biostats

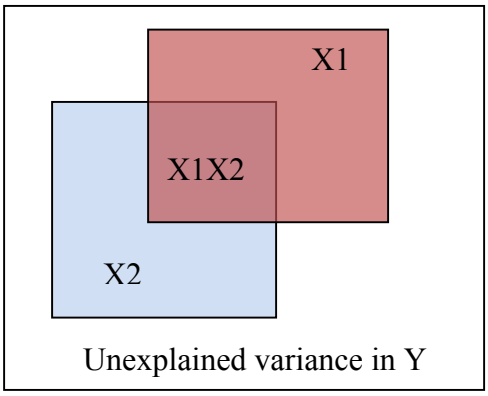
BACKGROUND

When the effects of a particular predictor dataset need to be tested after elimination of possible effects due to other variables, partial ordination may be used (e.g. partial CCA, partial RDA). Such an approach is also referred to as 'partialling out' or 'controlling for' the effects of specific variables, which are specified as covariables in the constrained analysis. In contrast to the previous exercise in which a single set of explanatory variables (or constraints) were used in the constrained ordination, here we are going to subdivide or partition the explanatory variables into logical subsets of variables and determine how much of the species variance can be explained independently by each subset or jointly by two or more subsets. Specifically, we seek to understand the independent and joint (or confounded) explanatory power of the logical subsets of variables.

The variance decompositions that result from canonical partitioning provide a comprehensive picture of the relative importance, independent effects, and confounding of the factors included in the analysis. The canonical partitioning method was originally used to partition variation in community data sets among environmental and spatial components (Borcard et al. 1992). Anderson and Gribble (1998) extended the technique to include the effects of temporal variation, so that variation in community data is partitioned into the components explainable by environmental, spatial, and temporal factors, and their overlap. Cushman and McGarigal (2002 and 2004) extended the method to address the specific challenges of hierarchically structured data.

Importantly, the canonical partitioning approach can be applied to a variety of ordination procedures, including the linear methods of redundancy analysis (RDA) and distance-based RDA (dbRDA) (also referred to as constrained analysis of principal coordinates, CAP), and the unimodal method of canonical correspondence analysis (CCA).

Before diving into the analyses, let's explore the general concept of partial CCA (or RDA or CAP) using a simple example. In doing so you will quickly see that partial direct ordination operates in a fashion similar to partial regression. In this example we have the species response matrix **Y** and two predictor matrices **X1** and **X2**. First, both sets of predictor matrices (**X1** and **X2**) are included in a constrained ordination. The amount of variance explained (total inertia for CCA) corresponds to the total variance explained by **X1** and **X2**. Next, a partial CCA is performed where the effect of **X1** is cleansed from the species matrix **Y**. CCA is performed using only **X1** and residuals from the analysis are submitted to a second CCA using **X2** as the constraining matrix. Effectively, the effect of **X1** is removed from the species community matrix **Y** and the unique proportion of



variance explained by **X2** can be determined. The process is repeated a second time, but now the species community data set is cleansed of the effects of **X2** in order to isolate the variance explained by **X1.** The figure above conceptualizes the process of variance partitioning, where **X1X2** is the proportion of variance explained by both matrices **X1** and **X2**.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset and environment dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
```

Now let's transform both datasets before diving into the analysis. First, let's conduct a 'row normalization' (i.e., to rescale each row so that the sums equal 1, or in other words to calculate relative species abundances), type the following:

```
speabu.tran<-data.stand(speabu,method='total',margin='row',plot=F)
```

Next, let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

PARTIAL CANONICAL CORRESPONDENCE ANALYSIS

Here, we will examine the independent and joint variation in species composition explained by environmental descriptors that depict natural habitat processes versus human-induced impacts on the river. Natural questions to address are whether natural and human variables explain unique or shared variance in the fish community data and whether one set of variables explains more variance than the other. To help with the analysis below, let's divide the environmental variables into 2 matrices by combining columns from envdata.tran using the function cbind(). Do this by typing:

```
attach(envdata.tran)
env.human<-cbind(HabQual, RoadDen,Agricult,HumanUse)
env.natural<-cbind(Sinuosity,Slope,WDRatio,SubEmbed,Elev,BasinAre)
detach(envdata.tran)
```

Conduct the variance partitioning

In this lab we will use two different ways of conducting pCCA in R – both of which utilize the cca function and thus result in the same output. The cca() function in the vegan library allows for both unconstrained correspondence analysis (CA) as well as the constrained version, canonical correspondence analysis (CCA). Recall that we used this function in the previous chapter. Its usage is:

cca (X, Y, Z,…)

Where:

13. **X** is the community data set
14. **Y** is the constraining or explanatory matrix
15. **Z** is a conditioning matrix (explained in more detail under "Partial CCA and RDA")

For additional documentation on **cca** type **?cca.**

To compute the variance partitioning where we isolate the variation in species composition explained uniquely by human factors (constrained component) we can type:

```
spe.pcca1<-cca(speabu.tran,env.human,env.natural)
summary(spe.pcca1)

Call:
cca(X = speabu.tran, Y = env.human, Z = env.natural)

Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total          6.3464    1.00000
Conditioned    1.4749    0.23240
Constrained    0.5219    0.08224
Unconstrained  4.3495    0.68536

Eigenvalues, and their contribution to the mean squared contingency
coefficient after removing the contribution of conditioning variables

Importance of components:
                        CCA1    CCA2    CCA3    CCA4    CA1     CA2     CA3
Eigenvalue            0.3113 0.10830 0.05718 0.04516 0.5258 0.48220 0.42660
Proportion Explained  0.0639 0.02223 0.01174 0.00927 0.1079 0.09898 0.08757
Cumulative Proportion 0.0639 0.08613 0.09787 0.10714 0.2151 0.31406 0.40163
```

From the summary above we see that the predictor matrix containing variables that describe human activities (i.e., env.human) uniquely explains 8.2% of variance in fish community composition across the stream sites.

Let's repeat the same variance partitioning, but this time we will isolate the variation in species composition explained uniquely by naturals factors. Notice that we simply switched the Y and Z matrices! Do this by typing:

```
spe.pcca2<-cca(speabu.tran,env.natural,env.human)
summary(spe.pcca2)

Call:
cca(X = speabu.tran, Y = env.natural, Z = env.human)

Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total          6.3464     1.0000
Conditioned    0.6622     0.1043
Constrained    1.3346     0.2103
Unconstrained  4.3495     0.6854

Eigenvalues, and their contribution to the mean squared contingency
coefficient after removing the contribution of conditioning variables

Importance of components:
                         CCA1    CCA2    CCA3    CCA4    CCA5    CCA6     CA1
Eigenvalue            0.39899 0.35739 0.23004 0.16525 0.09713 0.08579 0.52582
Proportion Explained  0.07019 0.06288 0.04047 0.02907 0.01709 0.01509 0.09251
Cumulative Proportion 0.07019 0.13307 0.17354 0.20261 0.21970 0.23479 0.32730
```

From the summary above we see that the predictor matrix containing variables that describe natural habitat characteristics (i.e., env.natural) uniquely explains 21.0% of variance in fish community composition across the stream sites. Based on either analysis, we can calculate the total variance explained by both env.human and env.natural by adding the conditional and constrained inertia in either table or by subtracting the unconstrained inertia from 1. This equals 31.4% of the total variation in fish community composition explained. Next, we can simply calculate the variance jointly explained by human and natural matrices as 31.4-(8.2+21.0) = 2.2%.

Perhaps a better way of conducting the above partitioning is to utilize the BIOSTATS library. To compute the variance partitioning, type:

```
spe.pcca<-ordi.part(speabu.tran,env.natural,env.human,method='cca')
```

In this function the first three arguments give the y-data set (species data) and the two explanatory data sets (x1, x2). Currently, the function does not accept more than three explanatory data sets. And while it is theoretically possible to include any number of explanatory data sets, the results become exceedingly difficult to interpret with more than three data sets. The next argument (method=) specifies the constrained ordination method to be used. Currently the function only accepts methods 'cca' and 'rda'. An additional optional argument (p='matrix name') which was not included above would provide you the opportunity to designate a "conditional" data set that is partialled out of all analyses. That is, its effects are removed from the community matrix, and the residual matrix is submitted to the next stage of the analysis. This approach can be used to remove the effect of some conditioning or covariables before the variance partitioning (say for example, space).

Interpreting the partitioning results

Below is the output that results from the command (ordi.part) conducted previously. Let's briefly decompose the output and figure out how to interpret the results in the process.

```
Call:
ordi.part(y = speabu.tran, x1 = env.natural, x2 = env.human, method = "cca")

Explanatory tables:
X1:  env.natural
X2:  env.human

Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total           6.346      1.000
Constrained     1.997      0.315
Unconstrained   4.350      0.685

Marginal effects:
   Inertia Prop.Total Prop.Constrained Pvalue
x1   1.475      0.232            0.739  0.005
x2   0.662      0.104            0.332  0.170

Components:
    Inertia Prop.Total Prop.Constrained Pvalue
v1    1.335      0.210            0.668  0.005
v2    0.522      0.082            0.261  0.470
v12   0.140      0.022            0.070     NA
```

Partitioning of variance - The summary reports a summary of the overall "unpartitioned" constrained ordination.

Total inertia (either 'variance' in RDA/CAP or the 'mean squared contingency coefficient" in CCA) is interpreted as the total "variance" in the species data set. This is the total "variance" in the species data that is partitionable into various components, both explained and unexplained.

Conditional inertia is the total "variance" in the species data set attributable to the conditional data set, if provided. This is the proportion of the species variance that is effectively removed before the subsequent partitioning into explained and unexplained components. This is not provided in the above analysis because we did not specify a conditional matrix.

Constrained inertia is the total "variance" in the species data set attributable to all of the explanatory variables or constraints; it is equal to the sum of the canonical eigenvalues. This is the so-called "explained variance".

Unconstrained inertia is the total "variance" in the species data set left unexplained or unaccounted for by the conditional and/or explanatory variables; it is equal to the sum of the unconstrained eigenvalues. This is the so-called "residual variance".

Marginal effects

Marginal effects represent the total "variance" in the species data attributable to a single explanatory data set or a combination of explanatory data sets, without partialling out any other explanatory data sets. This is the variance that can be attributed to a set of explanatory variables without consideration of potential confounding with other variables. The marginal effects are given as:

Total inertia attributed to the corresponding marginal variables (equal to the sum of the canonical eigenvalues). This is the marginal effect size.

Proportion of the total species variance (equal to the sum of the canonical eigenvalues divided by the total inertia). This is the relative effect size given in terms of the total species variance.

Proportion of the total constrained inertia (equal to the sum of the canonical eigenvalues divided by the total constrained inertia). This is the relative effect size given in terms of the total "explained" variance. This effect size can be quite large even though the percent of species variance explained is quite low.

Monte Carlo test of significance of the marginal effect size. This is a permutation test of the null hypothesis that the species are unrelated to the explanatory variables (with or without controlling for conditional variables). There are several optional arguments for controlling the permutation test (see help(anova.cca) for more details).

Components

Lastly, the summary reports the statistics for the partition components. Each component corresponds to a single exclusive (non-overlapping) partition of the total species variance, as can be represented in a Venn diagram (see below).

A two-way partitioning (involving two explanatory data sets) has three components:
x1 - pure x1 effects; the species variance explained due to x1 constraints alone, independent of the variance due to x2.
x2 - pure x2 effects; the species variance explained due to x2 constraints alone, independent of the variance due to x1.

x12 - joint x1-x2 effects; the species variance that is jointly explained by x1 and x2 constraints; i.e., the confounded variance that cannot exclusively be associated with either x1 or x2.

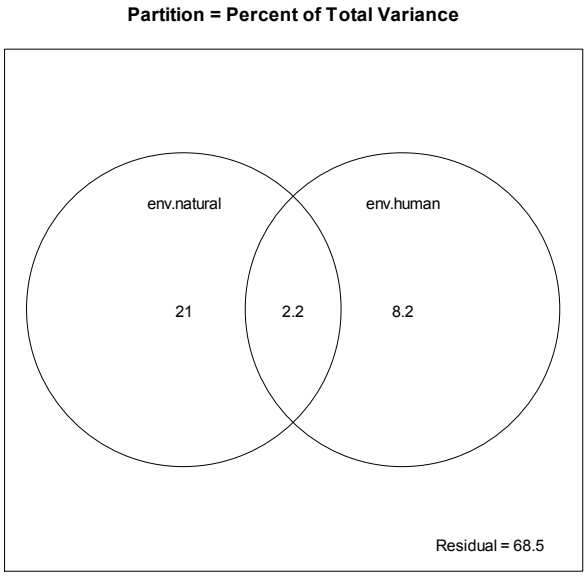
The third column reports the partition sizes in terms of the proportion of the total explained (or constrained) variance instead of the proportion of the total species variance.

In addition, the remaining "unexplained" variance is usually referred to as the "residual" variance and is equal to the sum of the unconstrained eigenvalues. If a "conditional" data set was provided, the explained variance, partitioned into the three components above, and the residual variance are based on the variance in the species data after accounting for that due to the conditional data set.

Venn Diagrams

The entire variance partitioning can be graphically portrayed in a Venn diagram using the plot.ordi.part() function in biostats, by typing:

```
plot.ordi.part(spe.pcca,which='total')
```



The Venn diagram depicts the components (from the summary above) of the two- or three-way partition, in addition to the so-called "residual" variance and "conditional" variance if provided. The components can be displayed either as the percentage of the total species variance accounted for (which='total') or the percentage of the total explained variance accounted for (which='constrained'), as follows:

```
plot.ordi.part(spe.pcca,which='constrained')
```

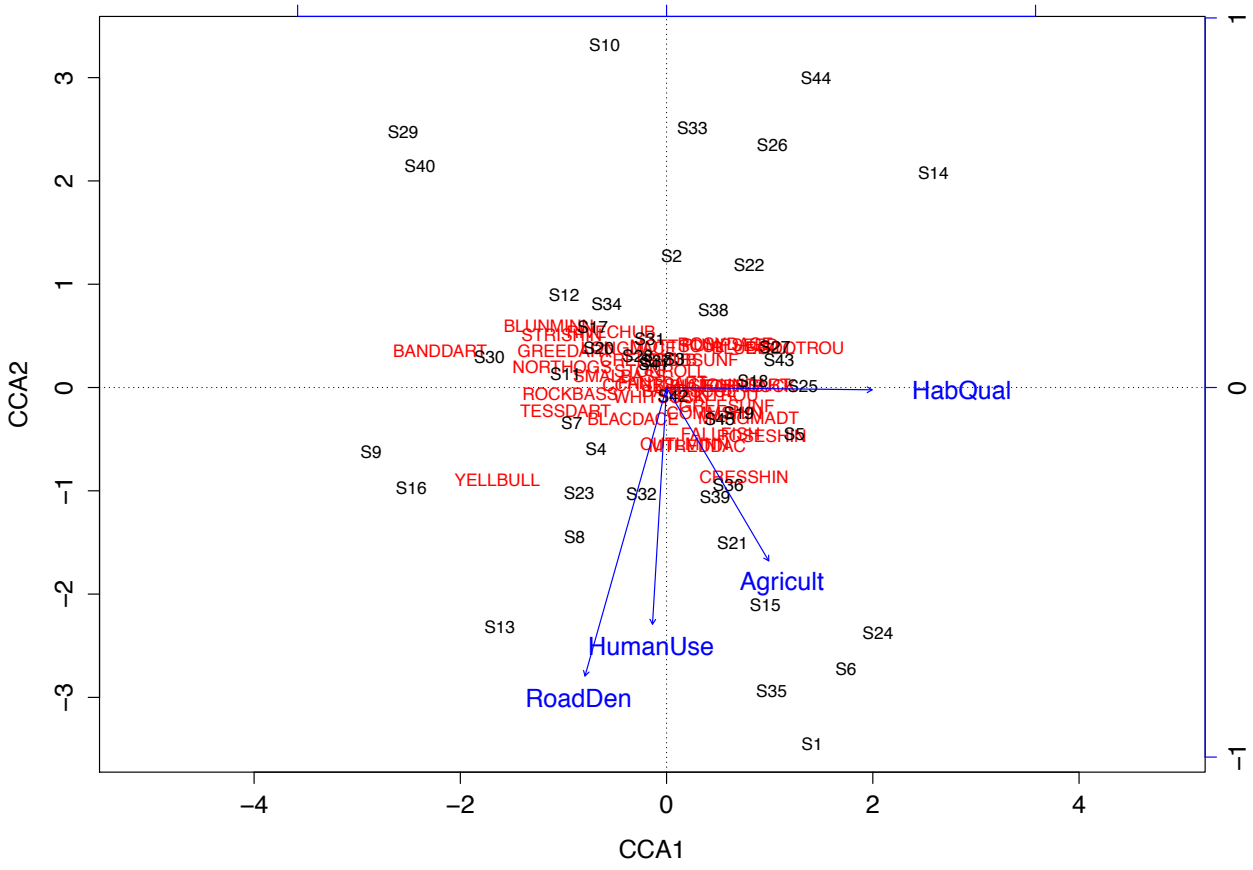
Partial triplots

The partitioning results can also be inspected using partial triplots to examine the relationships among samples, species and environmental variables in any particular partition. Partial triplots can be created for all of the marginal effects or any of the components of the partition that can be represented by a partial constrained ordination model. The latter includes the conditional effects of the explanatory variable subsets, but not the components representing the joint or overlapping regions. For example, if we are interested in examining the conditional human use partition; that is,

the independent effect of human use on species after controlling for the effects of natural habitat variables, we can type:

```
plot(spe.pcca1,choices=c(1,2),display=c('wa','sp','bp'),scaling=2)
```

Note that we are using the output from the cca() function and not ordi.part(). This is important because the ordi.part() function only reports the inertia values and does not contain the canonical scores and inter-/intra-correlation coefficients needed to create a triplot. The resulting triplot is below:



HIERARCHICAL VARIANCE PARTITIONING

If the explanatory variables are organized into a meaningful hierarchy, we can decompose each partition above into lower-level partitions. For example, we might be interested in decomposing the variance explained marginally or conditionally by each subset of variables into the independent and joint explanatory power of each variable. This can be accomplished using the same procedure as above, only adding additional variables to the conditional data set. I refer you to Cushman and McGarigal (2002) for a great review on hierarchical variance partitioning, including the mechanics of its calculation.

OPTIONAL READINGS

Anderson, M.J. and N.A. Gribble. 1998. Partitioning the variation among spatial, temporal and environmental component in a multivariate data set. Aust. J. Ecol. 23: 158–167.

Borcard D., Legendre P., and P. Drapeau. 1992. Partialling out the spatial component of ecological variation. Ecology 73: 1045–1055.

Cushman, S. A., and K. McGarigal. 2002. Hierarchical, multi-scale decomposition of species-environment relationships. Landscape Ecology 17:637-646.

Meot, A., P. Legendre and D. Boncard. 1998. Partialling out the spatial component of ecological variation: questions and propositions in the linear modelling framework. Environmental and Ecological Statistics 5:1-27.

13. Linear Discriminant Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Linear Discriminant Analysis (LDA)

R Packages: vegan, mass, caret, e1071
R Source: biostats

BACKGROUND

Linear discriminant analysis (LDA) is an ordination technique that is mathematically similar to CCA (see lecture notes and past chapter), but generates canonical variates (CV), or linear combinations, of the descriptors in a manner that reduce within-group dissimilarity and maximized between-group dissimilarity. The resulting canonical variates capture gradients that emphasize group differences. As with CCA, the descriptor loadings for each CV can be examined; high loadings indicate descriptors that are important for assigning objects into groups.

SET-UP

In this exercise you will be working with the MAHA species environmental dataset and site group dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan, mass, caret and e1071 Packages. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
sitegroup <- read.csv('MAHA_groups.csv',header=TRUE, row.names=1)
```

Recall that **`sitegroup`** contains two column vectors depicting the major basin and degree of human impact for each site (object).

Let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

LINEAR DISCRIMINANT ANALYSIS

For today we'll be performing LDA on the class dataset to examine environmental differences between streams belong to different river basins. We will use the function **lda**() from the MASS library.

Its usage is:

lda(x, grouping)

Where:

- **x** is a matrix containing the explanatory variables
- **grouping** is a factor specifying the class for each object

Now, let's perform LDA by typing:

```
env.lda<-lda(envdata.tran, sitegroup[,1])
```

To return the results from the analysis, type:

```
env.lda
```

```
Call:
lda(envdata.tran, sitegroup[, 1])

Prior probabilities of groups:
    MidAt      Ohio   SouthAt   Tenness
0.5111111 0.2444444 0.1333333 0.1111111

Group means:
        Sinuosity     Slope  WDRatio SubEmbed  HabQual     Elev  RoadDen
MidAt   0.3288261 0.3669565 1.278652 1.706826 1.152217 2.974522 1.133391
Ohio    0.3246364 0.3814545 1.274364 1.664545 1.155091 3.301364 1.106364
SouthAt 0.3326667 0.2485000 1.329000 1.787167 1.172333 2.921000 1.234167
Tenness 0.3576000 0.4122000 1.478000 1.824200 1.164200 3.229600 0.945800
         Agricult HumanUse BasinAre
MidAt   0.9588261 1.103696 1.176087
Ohio    0.9611818 1.104727 1.377909
SouthAt 1.4280000 1.467833 1.484333
Tenness 0.9960000 0.996000 1.439200

Coefficients of linear discriminants:
                 LD1        LD2        LD3
Sinuosity 19.5401523 23.6620108  6.2305313
Slope     -0.2621492  0.7241286  1.3955106
WDRatio   -2.5921886  6.6425517 -1.4506424
SubEmbed   1.1510945  3.3176814 -0.8793224
HabQual   -6.5656873 -1.6136564 -0.8644764
Elev       5.0044865 -0.6176585 -1.7183475
RoadDen   -3.9206046  0.3713147 -4.1245479
Agricult  -2.9994417  1.0659918 -1.9474969
HumanUse   3.1808550 -1.0933945  2.0399924
BasinAre  -0.1725865 -0.9891639 -0.7316773

Proportion of trace:
   LD1    LD2    LD3
0.5351 0.3807 0.0842
```

The resulting object is a list containing a number of components, and the call to the object prints several of these components, including the prior probabilities of group membership (assumed by default to be proportional to group sample sizes), the group means on each variable, the coefficients of the canonical linear discriminant functions, and the trace values (related to eigenvalues).

The coefficients of linear discriminants indicate the variables that contribute the most towards separating groups along each axis: for LD1 (the first canonical variate), sinuosity and habitat quality had the highest contribution while for LD2, sinuosity and width:depth ratio were the most important. The proportion of trace represents the proportion of variation (in the grouping variable) that is

described by each of the canonical variates (LDs). Note that the number of canonical variates is equal to 1 minus the number of groups.

Canonical Scores

We can compute the canonical scores, i.e., the scores for each sample observation on each canonical axis, using the predict() function, as follows:

```
env.lda.pred<-predict(env.lda)

env.lda.pred
```

The following summary will be produced.

```
$class
 [1] MidAt   MidAt   MidAt   MidAt   MidAt   SouthAt Ohio    Ohio    Ohio
[10] MidAt   MidAt   Tenness MidAt   MidAt   MidAt   MidAt   MidAt   SouthAt
[19] MidAt   Ohio    MidAt   MidAt   MidAt   MidAt   MidAt   MidAt   Ohio
[28] Ohio    MidAt   Ohio    Ohio    Ohio    MidAt   MidAt   MidAt   MidAt
[37] Tenness Ohio    MidAt   Tenness MidAt   Ohio    MidAt   MidAt   MidAt
Levels: MidAt Ohio SouthAt Tenness

$posterior
           MidAt         Ohio      SouthAt      Tenness
S1  0.5334109098 0.0681528881 0.3770980104 2.133819e-02
S2  0.9433982146 0.0152093499 0.0412897967 1.026388e-04
S3  0.5948130798 0.0092860567 0.3914960420 4.404821e-03
S4  0.7166114839 0.2709046543 0.0055784840 6.905378e-03
S5  0.5479636054 0.0037339738 0.4466555938 1.646827e-03…

$x
           LD1          LD2          LD3
S1  -0.54170547  0.582602803 -1.364877650
S2  -1.50441412 -1.074784221  1.609699485
S3  -1.48085012  0.650769018 -0.320353095
S4   0.62911095 -0.600231804  1.355820928
S5  -1.94349946  0.609121242 -0.081822334
…
```

Note, the resulting object contains the predicted group membership (which we will use below to evaluate the classification criterion), the posterior probability of membership in each group (which is the basis for the prediction of group membership), and the canonical scores. For convenience, we can extract the scores from this object (the list component named 'x') and store them in a separate object, by typing:

```
scores<-env.lda.pred$x
```

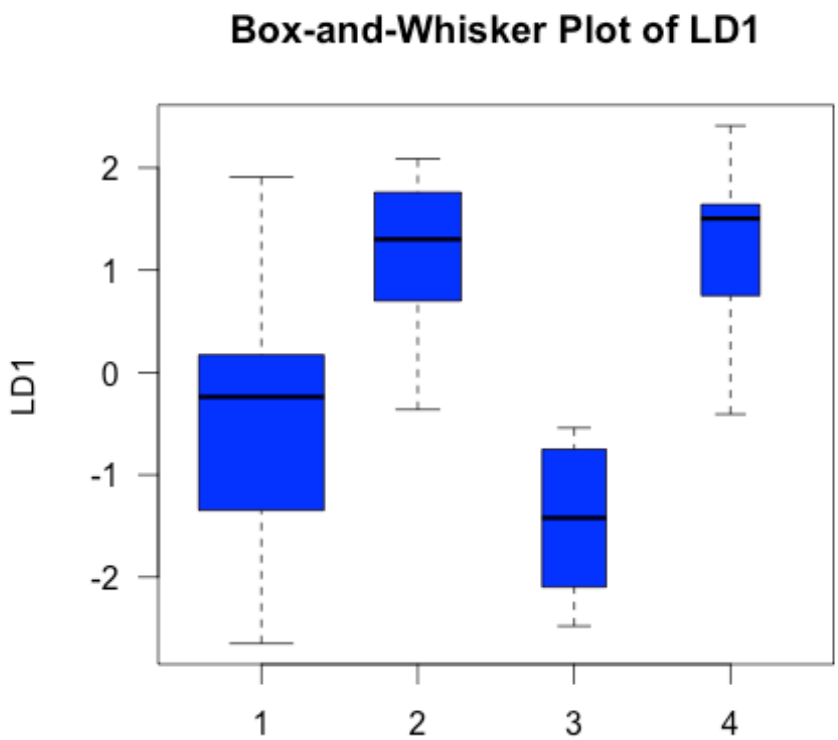
Next, most find it quite useful to graphically examine the canonical separation of groups to get a sense of how well the LDA worked. Plots of the canonical scores graphically illustrate the relationships among objects, because objects in close proximity in canonical space are ecologically similar with respect to the environmental gradients defined by the canonical functions, and are commonly used to assess how much overlap exists in group distributions; i.e., how distinct the groups are.

To use the box.plots() and hist.plots() convenience functions biostats, we need to bind the grouping variable (sitegroup[,1]) to the data frame containing the scores (scores), by typing:

```
result<-as.data.frame(cbind(sitegroup[,1],scores))
```

Then we can produce grouped box-and-whisker plots by typing:

```
box.plots(result,var='LD1',by='V1',notch=TRUE,varwidth=TRUE)
```



Classification Success

We can measure the accuracy of the classification criterion to indirectly assess the amount of canonical discrimination contained in the variables. The higher the correct classification rate, the greater the degree of group discrimination achieved by the canonical functions. Note, classification, that is, the predictive side of DA, is often the intended purpose of the analysis and may be pursued independent of the canonical analysis of discriminance. Here, we will use classification in both ways: to help assess how well the canonical discrimination worked and as a possible criterion for classifying future observations.

Recall from above that we already used the predict() function on the lda object to get the canonical scores. The predict object also contains the predicted group (or class) membership of each observation based on the posterior probabilities of group membership in each group. The predict() function includes a few different options for calculating the posterior probabilities of group membership, but we will not concern ourselves with the differences among methods here. We will simply use the default method for this exercise. The predicted group membership information is stored in the predict object in the component named 'class'. We can summarize the group predictions in a confusion matrix using the table() function, as follows:

```
env.table<-table(sitegroup[,1],env.lda.pred$class)
env.table
```

```
          MidAt Ohio SouthAt Tenness
  MidAt      21    2       0       0
  Ohio        2    9       0       0
  SouthAt     4    0       2       0
  Tenness     2    0       0       3
```

The rows of the table are the 'from' group (i.e. the true or known group of each observation) and the columns are the 'to' group (i.e., the predicted group membership). The diagonals of the table give the correct predictions (frequencies) and the off-diagonals give the errors of omission and commission. To compute the overall correct classification rate (CCR), type:

```
sum(diag(env.table))/sum(env.table)
```

```
0.7777778
```

This shows that the LDA model correctly classifies sites

Better yet, let's calculate a whole suite of indices based on this table (which is called a confusion matrix), including measures of CCR, sensitivity, specificity, Cohen's Kappa index, etc., by typing:

```
confusionMatrix(env.table)
```

Interpreting the Canonical Functions

Assuming that we achieve good group separation on the canonical axes, our next task is to determine if we can provide a meaningful ecological interpretation of the group differences. First, we might suspect that the eigenvector coefficients or the canonical coefficients of the linear discriminant functions might provide the answer. We can view the raw canonical coefficients of the linear functions by typing:

```
env.lda$scaling
```

Note, these are the coefficients to be applied to the raw, unstandardized variables to produce unstandardized canonical scores and are generally uninterpretable as coefficients. To aid in the interpretation of the canonical functions, it is generally preferable to compute the canonical structure correlations. These are Pearson product-moment bivariate correlations between each discriminating variable and each canonical function (i.e., the canonical scores associated with each axis). As such, structure coefficients generally are not affected by relationships with other variables. These structure coefficients can be computed using the lda.structure() function in biostats, by typing:

```
lda.structure(scores,envdata.tran)
```

```
Structure Correlations:
             LD1    LD2    LD3
Sinuosity  0.096  0.484 -0.026
Slope      0.293  0.023  0.453
WDRatio    0.143  0.565 -0.209
SubEmbed  -0.085  0.343 -0.174
HabQual   -0.031  0.076 -0.206
Elev       0.716 -0.074 -0.401
RoadDen   -0.336 -0.265 -0.252
Agricult   -0.19  0.079 -0.489
HumanUse  -0.196 -0.025 -0.395
```

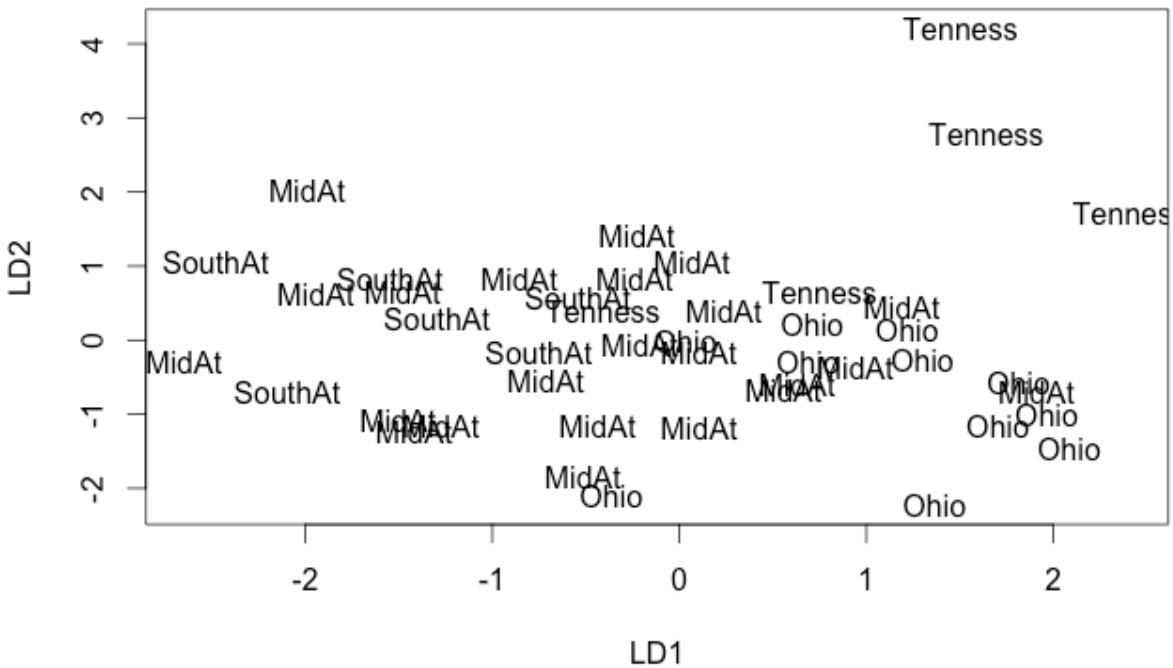
```
BasinAre    0.09  0.117   -0.6
```

The structure correlations indicate the strength and nature (positive or negative) of the relationship between each discriminating variable and each canonical axis and can be used to interpret the ecological meaning of each axis. We can define a canonical function on the basis of the structure coefficients by noting the variables that have the largest loadings. And the squared loadings indicate the percent of the variable's variance accounted for by that function.

LDA Bioplot

Finally, let's visualize the ordination of the canonical axes from the LDA.

```
plot(scores,type='n')
text(scores,labels=sitegroup[,1])
```



14. Classification and Regression Trees

FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- Classification and Regression Trees (CART)

R Packages: rpart, rpart.plot
R Source: None

BACKGROUND

Modeling ecological data poses many challenges. The response as well as the explanatory variables may be continuous or discrete. The relationships that need to be deciphered are often nonlinear and involve complex interactions between explanatory variables. Missing values for both explanatory and response variables are not uncommon, and outliers almost always exist. In addition, ecological problems usually demand methods that are both easy to implement and easy to interpret. Frequently, many different statistical tools are employed to handle unique problems posed by the various scenarios. Classification and regression trees (Breiman et al. 1984), however, offer a single tool to work with all these challenges.

CART analysis is a form of binary recursive partitioning where classification and regression trees refer to the modeling of categorical and continuous response variables, respectively. The term "binary" implies that each group of observations, represented by a node in a decision tree, is split into two child nodes, a process through which the original node becomes a parent node. The term "recursive" refers to the fact that the binary partitioning process can be applied repetitively. Thus, each parent node can give rise to two child nodes and, in turn, each of these child nodes may themselves be split, forming additional children. The term "partitioning" refers to the fact that the dataset is split into sections or partitioned. Although there are many different versions of binary recursive partitioning available the overall methodology is consistent regardless of the exact implementation. See Olden et al. (2008).

CART analysis consists of three basic steps. The first step involves tree building, during which a decision tree is built by repeatedly partitioning the data set into a nested series of mutually exclusive groups, each of them as homogeneous as possible with respect to the response variable. Tree building begins at the root node with the entire dataset, and the algorithm formulates split-defining conditions for each possible value of all the independent variables to create candidate—or surrogate—splits. Other splitting criteria are also available. Next, the algorithm selects the best candidate split that minimizes the average "impurity" of the two child nodes. Impurity is based on a goodness of fit measure, such as the information (entropy) index and the Gini index for classification trees and sums of squares about group means for regression trees (De'ath and Fabricius 2000). The algorithm continues recursively with each of the new children nodes until tree building is stopped. The second step consists of stopping the tree building process. The process is stopped when: (1) there are only n observations in each of the child nodes (where n is set by the user), (2) all observations within each child node have the identical distribution of independent variables, making splitting impossible, or (3) an external limit on the number of splits in the tree or minimum purity threshold is achieved. A terminal node or "leaf" is a node that the algorithm cannot partition any further because one of the above criteria is met. In classification trees, each node—even the root node—is assigned a predicted probability for each class (often the class with the greatest probability is assigned to the node). For regression trees, the predicted value for each node is typically defined as the mean or median value of the response variable for the observations in the node. The third step involves tree pruning and optimal tree selection. It is common practice to determine the optimal size of the decision tree (i.e., the number of terminal nodes) by selecting the tree size with the smallest model error based on repeated cross-validations of the data. Alternatively, the smallest tree whose model error falls within the one standard error rule is also used (see De'ath and Fabricius 2000).

Based on the branching topology of the decision tree, one can interpret the primary splits that represent the most important variables in the prediction process, as well as the best competitive surrogate splits that also show high classification power. To calculate the overall importance of the independent variables in a decision tree, the CART analysis quantifies the improvement measure attributable to each variable in its role as a surrogate to the primary split. The values of these improvements are summed over each node, totaled, and scaled relative to the best performing variable, i.e., expressed as a relative importance on a 0–100% scale (Breiman et al. 1984).

SET-UP

In this exercise you will be working with the MAHA species presence-absence dataset, and environmental dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the rpart and rpart.plot Packages.

Let's import the data, by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
```

Next, let's calculate the total number of fish species at each site (richness) so that we have a single continuous descriptor of fish community composition.

```
sperich<-rowSums(speocc)
```

CART

For today we'll be performing CART on the class dataset to examine the relationship between fish species richness and environmental differences between streams. We will use the function **rpart**() from the rpart library.

Its usage is:

rpart(formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)

Where:

- **`formula`** defines the reponse and predictor variables
- **`data`** is the dataframe from which the variables are identified
- **`method`** is one of "anova", "poisson", "class" or "exp". If method is missing then the routine tries to make an intelligent guess. If y is a survival object, then method = "exp" is assumed, if y has 2 columns then method = "poisson" is assumed, if y is a factor then method = "class" is assumed, otherwise method = "anova" is assumed.

Now, let's perform a regression tree by typing:

```
tree<-rpart(sperich~Slope+HabQual+Elev+RoadDen+Agricult+HumanUse+BasinAre,
envdata,method="anova",minsplit=5,xval=3)
```

In this case, minsplit and xval are optional parameters (type **?rpart.control** for more information). Minsplit refers to the the minimum number of observations that must exist in a node in order for a split to be attempted. Xval refers to the number of cross-validations to determine the pruned tree.

Next, let's explore the regression tree by typing:

summary(tree)

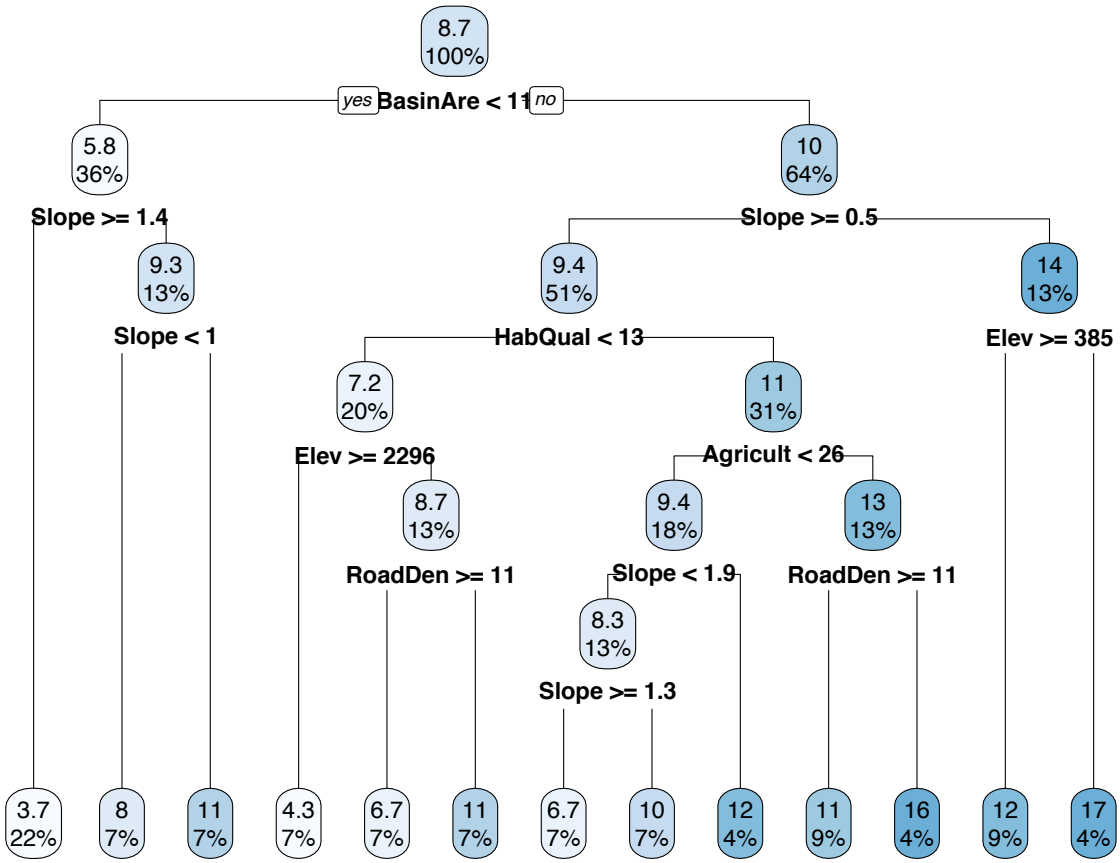
The following is produced:

```
Call:
rpart(formula = sperich ~ Slope + HabQual + Elev + RoadDen +
    Agricult + HumanUse + BasinAre, data = envdata, method = "anova",
    minsplit = 5, xval = 3)
  n= 45

           CP nsplit  rel error   xerror      xstd
1  0.27337478      0 1.00000000 1.088625 0.1968294
2  0.15359340      1 0.72662522 1.225393 0.2382156
3  0.13045104      2 0.57303182 1.378458 0.2671437
4  0.08978519      3 0.44258079 1.378458 0.2671437
5  0.04847129      4 0.35279560 1.375599 0.2645525
6  0.04794632      5 0.30432431 1.552375 0.3104226
7  0.03484770      6 0.25637799 1.658015 0.3232672
8  0.03361082      7 0.22153029 1.706002 0.3160573
9  0.03108329      8 0.18791946 1.706002 0.3160573
10 0.03097574      9 0.15683617 1.706002 0.3160573
11 0.02151093     10 0.12586044 1.706002 0.3160573
12 0.01376699     11 0.10434951 1.706002 0.3160573
13 0.01000000     12 0.09058252 1.754259 0.3365014

Variable importance
   Slope BasinAre     Elev Agricult  HabQual  RoadDen HumanUse
      26       22       12       11       11       11        7

Node number 1: 45 observations,    complexity param=0.2733748
  mean=8.733333, MSE=17.21778
  left son=2 (16 obs) right son=3 (29 obs)
  Primary splits:
      BasinAre < 11.1073  to the left,  improve=0.2733748, (0 missing)
      Slope    < 1.6875   to the right, improve=0.2498709, (0 missing)
      Agricult < 0.145    to the left,  improve=0.1403854, (0 missing)
      Elev     < 490      to the right, improve=0.1141687, (0 missing)
      RoadDen  < 7.75     to the left,  improve=0.1084796, (0 missing)
  Surrogate splits:
      Slope    < 1.8925   to the right, agree=0.733, adj=0.250, (0 split)
      Agricult < 76.725   to the right, agree=0.689, adj=0.125, (0 split)
      HumanUse < 2.44     to the left,  agree=0.689, adj=0.125, (0 split)
      RoadDen  < 6.66     to the left,  agree=0.667, adj=0.062, (0 split)
```

This first table contains complexity parameters and relative error for difference tree sizes (i.e., number of splits or nsplit). Next, overall variable importance is provided. Finally, the primary and surrogate splits are provided for each split in the tree, as well as the predicted value of the response variable; in this case 8.73 species. Only the first one (root node) is shown above.

Next, let's visualize the regression tree by typing:

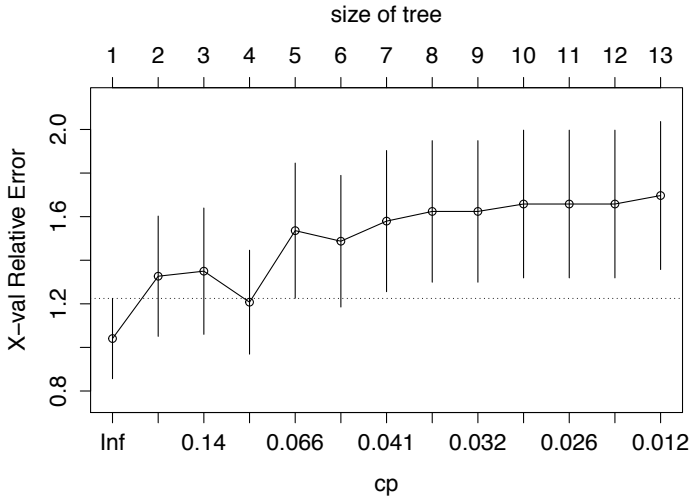
rpart.plot(tree)



Here, we are looking at the minimum cross-validated error tree. An alternative is to use the smallest tree that is within 1 standard error of the best tree (i.e., the one above). The reason for this is that, given the CV estimates of the error, the smallest tree within 1 standard error is doing just as good a job at prediction as the best (lowest CV error) tree, yet it is doing it with fewer "terms".

Let's first plot the cost-complexity vs tree size for the un-pruned tree by typing:

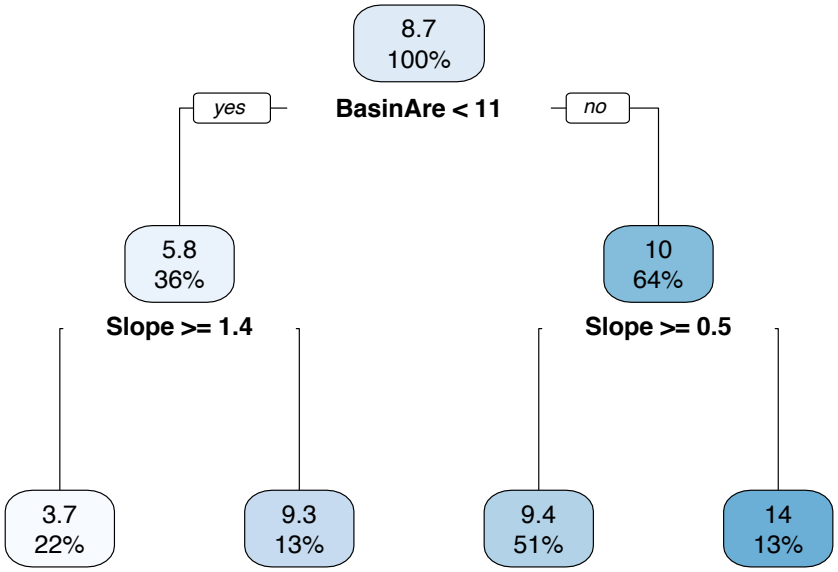
```
plotcp(tree)
```



A good choice of cp for pruning is often the leftmost value for which the mean lies significantly below the horizontal line representing the 1 SE of the minimum. In this case, cp=0.11).

Now, let's prune and plot the tree according to this cp value.

```
ptree<-prune(tree,0.11)
rpart.plot(ptree)
```



OPTIONAL READINGS (* recommended)

Breiman, L., J. H. Friedman, R. A. Olshen, and C. G. Stone. 1984. Classification and regression trees. Wadsworth International Group, Belmont, California, USA.

De'ath G and Fabricius KE (2000) Classification and regression trees: A powerful yet simple technique for ecological data analysis. Ecology 81: 3178–3192.

De'ath, G. 2002. Multivariate regression trees: a new technique for constrained classification analysis. Ecology 83:1103–1117.

Olden, J.D., Lawler, J.J., and N.L. Poff. 2008. Machine learning methods without tears: a primer for ecologists. The Quarterly Review in Biology 83: 171-193.

15. Multi-response permutation procedure (MRPP)

FISH 560: Applied Multivariate Statistics for Ecologists

R Packages: vegan
R Source: biostats

BACKGROUND

Multi-response permutation procedure (MRPP) is a nonparametric approach for testing the hypothesis of no difference between two or more groups of entities (Mielke et al. 1976). The strategy of MRPP is to compare the observed intra-group average distances with the average distances that would have resulted from all the other possible combinations of the data under the null hypothesis. The test statistic, usually symbolized with a lower case delta, is the average of the observed intra-group distances weighted by relative group size. The observed delta is compared to the possible deltas resulting from Monte Carlo permutations of the assignment of sample observations to the groups. If the hypothesis that the groups are not different (the null hypothesis) is true, then each of the possible assignments (permutations) is equally likely. The flexibility of MRPP stems from the use of any distance measure.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset and site group dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan and mass library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
sitegroup <- read.csv('MAHA_Groups.csv',header=TRUE, row.names=1)
```

Recall that **`sitegroup`** contains three column vectors depicting the major basin, degree of human impact and a column of text colors (which will help for graphing later) for each site (object).

MULTIPLE RESPONSE PERMUTATION PROCEDURE (MRPP)

Calculate dissimilar matrix
The first step in MRPP is the selection of a distance matrix to describe similarities among objects according to the descriptors. Again, any (dis) similarity coefficient can be selected. For today, let's generate a distance matrix using an appropriate dissimilarity coefficient for species abundance – Bray-Curtis coefficient. We'll use the vegdist() function (see Ecological Resemblace chapter). Type,

```
speabu.d<-vegdist(speabu, "bray")
```

Conduct MRPP
For MRPP, we can use the function **mrpp()** found in the vegan library (see usage below) or the **mrpp2()** function in BIOSTATS, which is simply an enhanced version of mrpp().

```
mrpp(dat, grouping, permutations = 999, distance = "euclidean",
     weight.type = 1, strata)
```

Where:

- dat data matrix or data frame in which rows are samples and columns are response variable(s), or a dissimilarity object or a symmetric square matrix of dissimilarities.
- grouping Factor or numeric index for grouping observations
- permutations Number of permutations to assess the significance of the MRPP statistic, delta.
- distance Choice of distance metric that measures the dissimilarity between two observations . See vegdist for options. This will be used if dat was not a dissimilarity structure of a symmetric square matrix.
- weight.type choice of group weights. See Details below for options.
- strata An integer vector or factor specifying the strata for permutation. If supplied, observations are permuted only within the specified strata.
- dist A dist object of dissimilarities, such as produced by functions dist, vegdist or designdist.

For more details and documentation type **?mrpp**.

Type:

```
sp.mrpp<-mrpp(speabu.d,sitegroup[,1])

sp.mrpp

Call:
mrpp(dat = speabu.d, grouping = sitegroup[, 1])

Dissimilarity index: bray
Weights for groups:  n

Class means and counts:

      MidAt  Ohio   SouthAt Tenness
delta 0.8199 0.8181 0.8551  0.8424
n     23     11     6       5

Chance corrected within-group agreement A: 0.03024
Based on observed delta 0.8267 and expected delta 0.8524

Significance of delta: 0.004
Based on  999  permutations
```

The resulting object is a list containing several components, but the most important information is the value of the test statistic, delta, the p-value derived from the Monte Carlo permutation test, and the value of the 'effect size', A. From the above output we can see that the river basins are significantly different in their fish community composition. If several groups are included in MRPP, subsequent paired tests can be performed to identify which groups in particular significantly differ.

Now type:

```
sp.md <- meandist(speabu.d, sitegroup[,1])
sp.md
```

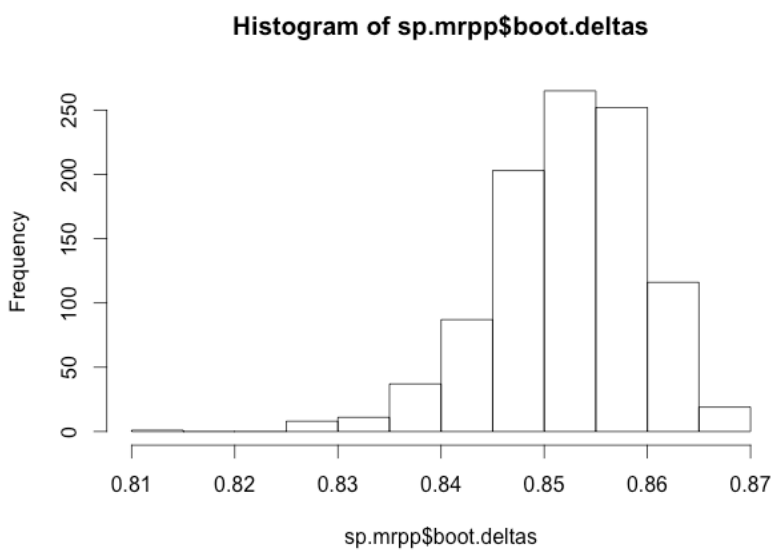
```
            MidAt      Ohio   SouthAt   Tenness
MidAt   0.8199154 0.8381517 0.8945901 0.8766627
Ohio    0.8381517 0.8180957 0.9151368 0.8236455
SouthAt 0.8945901 0.9151368 0.8551026 0.9404506
Tenness 0.8766627 0.8236455 0.9404506 0.8424387
attr(,"class")
[1] "meandist" "matrix"
attr(,"n")
grouping
  MidAt    Ohio SouthAt Tenness
     23      11       6       5
```

summary(sp.md)

```
Mean distances:
                 Average
within groups  0.8218763
between groups 0.8679377
overall        0.8524443

Summary statistics:
                          Statistic
MRPP A weights n         0.03024184
MRPP A weights n-1       0.03106854
MRPP A weights n(n-1)    0.03585929
Classification strength 0.04127288
```

We can plot the distribution of random delta values by typing:

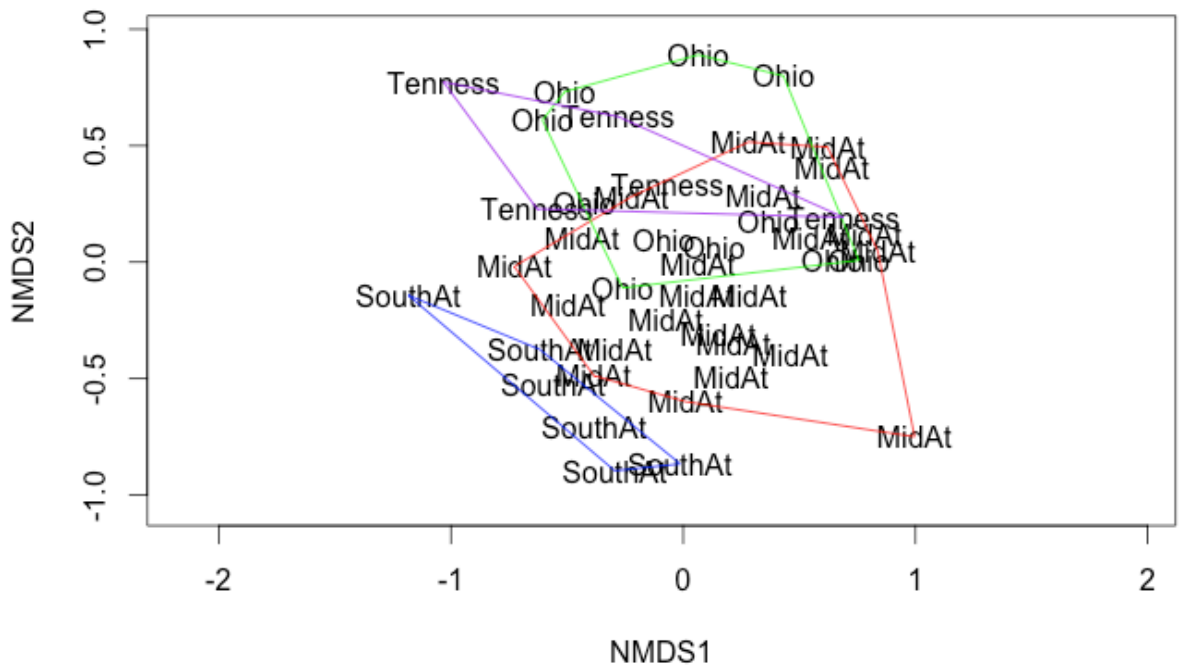
hist(sp.mrpp$boot.deltas)



Compare this null distribution to the observed delta of 0.8267.

To visualize group overlap in ordinate space, we can call the **ordihull ()** function to outline the extent of objects belonging to each group in ordinate space created from a NMDS. We also use different colors for each group. To do this type:

```
spe.nmds<-metaMDS(speabu, distance='bray', k=2, autotransform=FALSE,
trymax=100)

plot(spe.nmds,type='n')
text(spe.nmds,labels=sitegroup[,1])
ordihull(spe.nmds,sitegroup[,1],show.groups="MidAt",col=c("red"))
ordihull(spe.nmds,sitegroup[,1],show.groups="SouthAt",col=c("blue"))
ordihull(spe.nmds,sitegroup[,1],show.groups="Ohio",col=c("green"))
ordihull(spe.nmds,sitegroup[,1],show.groups="Tenness",col=c("purple"))
```



Finally, to produce a plot without different group colors simply type:

```
plot(spe.nmds,type='n')
text(spe.nmds,labels=sitegroup[,1])
ordihull(spe.nmds,sitegroup[,1])
```

OPTIONAL READINGS

Mielke, P. W., Jr., K. J. Berry, and E. S. Johnson. 1976. Multiresponse permutation procedures for a priori classifications. Communications in Statistics Theory and Methods **5**:1409–1424.

16. Analysis of Similarity (ANOSIM)

FISH 560: Applied Multivariate Statistics for Ecologists

R Packages: vegan
R Source: biostats

BACKGROUND

Ecologists are often interested in identifying factors that help explain the differences between two or more groups of sampling entities. The purpose might be to identify ecological or environmental factors that best explain differences in the distribution patterns of one or more organisms. Here, the emphasis is on explaining the differences among pre-specified, well-defined groups of sampling entities based on a suite of discriminating variables. Alternatively, the purpose might be to predict group membership for samples of unknown membership. For example, this might involve predicting whether a species is likely to occur at a site based on a suite of ecological or environmental variables. Here, the emphasis is on prediction.

ANOSIM (Analysis of Similarity) is a non-parametric multivariate procedure broadly analogous to ANOVA that has been widely used for testing whether or not groups of objects are statistically different in respect to their relative similarities according to a set of descriptors. It was originally proposed by Clarke (1993) to perform community analysis. ANOSIM tests priori-defined groups against random groups in ordination space by calculating the average of all rank similarities among objects within groups, and the average of rank similarities among objects between groups. A test statistic, R, lies between -1 and +1. A value of 1 indicates that all objects within groups are more similar to one another than any objects from different groups, a value of 0 indicates that there is no difference among groups (i.e., representing the null hypothesis), and a value of -1 indicates that all objects within groups are less similar to one another than any objects from different groups (Clarke and Gorley 2001).

SET-UP

In this exercise you will be working with the MAHA species abundance dataset and site group dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan and mass library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
sitegroup <- read.csv('MAHA_Groups.csv',header=TRUE, row.names=1)
```

ANALYSIS OF SIMILARITY (ANOSIM)

We can breakdown ANOSIM into a series of steps. Below, we tackle each step in turn.

Calculate dissimilar matrix
The first step in ANOSIM is the selection of a distance matrix to describe similarities among objects according to the descriptors. Again, any (dis) similarity coefficient can be selected. For today, let's generate a distance matrix using an appropriate dissimilarity coefficient for species abundance – Bray-Curtis coefficient. We'll use the vegdist() function (see Ecological Resemblace chapter).

Conduct the test of group differences
We will use the anosim() function in the vegan library to test the null hypothesis of no group differences. It's usage is:

anosim(dat, grouping, permutations = 999, distance = "bray", strata)

Where:

- dat Data matrix or data frame in which rows are samples and columns are response variable(s), or a dissimilarity object or a symmetric square matrix of dissimilarities.
- grouping Factor for grouping observations.
- permutations Number of permutation to assess the significance of the ANOSIM statistic.
- distance Choice of distance metric that measures the dissimilarity between two observations . See vegdist for options. This will be used if dat was not a dissimilarity structure or a symmetric square matrix.
- strata An integer vector or factor specifying the strata for permutation. If supplied, observations are permuted only within the specified strata.

```
y.anosim<-anosim(speabu.d,sitegroup[,1])
```

You will note that I supplied a distance matrix to the function rather than using the dat and distance arguments. Both will produce the same result, however, I wanted to illustrate the full flexibility of the function. The anosim() function contains a default summary() function which displays a bit more information from the analysis. Type:

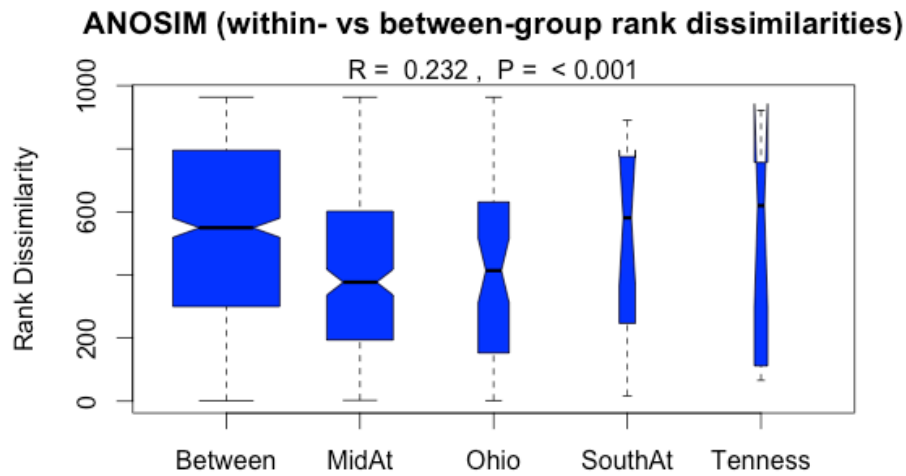
```
summary(y.anosim)

Call:
anosim(dat = speabu.d, grouping = sitegroup[, 1])
Dissimilarity: bray

ANOSIM statistic R: 0.2316
      Significance: 0.001

Based on  999  permutations

Upper quantiles of permutations (null model):
   90%    95%  97.5%    99%
0.0835 0.1100 0.1368 0.1609

Dissimilarity ranks between and within classes:
        0% 25%    50%     75%  100%   N
Between  1 300 549.50 795.000 963.5 657
MidAt    3 194 377.00 602.000 963.5 253
Ohio     2 153 414.00 631.000 963.5  55
SouthAt 17 247 581.00 775.000 890.0  15
Tenness 66 145 619.75 741.875 922.0  10
```

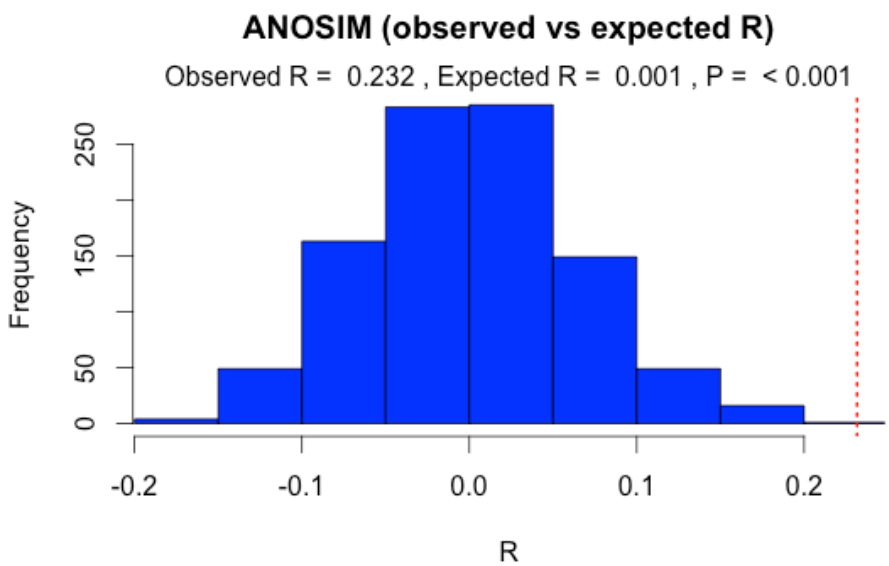
The most important information in the summary is the value of the test statistic, R, and the p-value derived from the Monte Carlo permutation test. Based on this information, can we reject the null hypothesis of no differences in fish community composition among the major river drainages? If so, is the difference between groups ecological significant as well based on the magnitude of R?

To facilitate interpretation, we can plot the dissimilarities in a grouped box-and-whisker plot using the plot.anosim() function in BIOSTATS, as follows:

```
plot.anosim(y.anosim)
```

This produces a box-plot illustrating the within- versus between-group dissimilarities, and the null distribution of R values (and observed R) in which the significance of the difference between groups is assessed (see below).



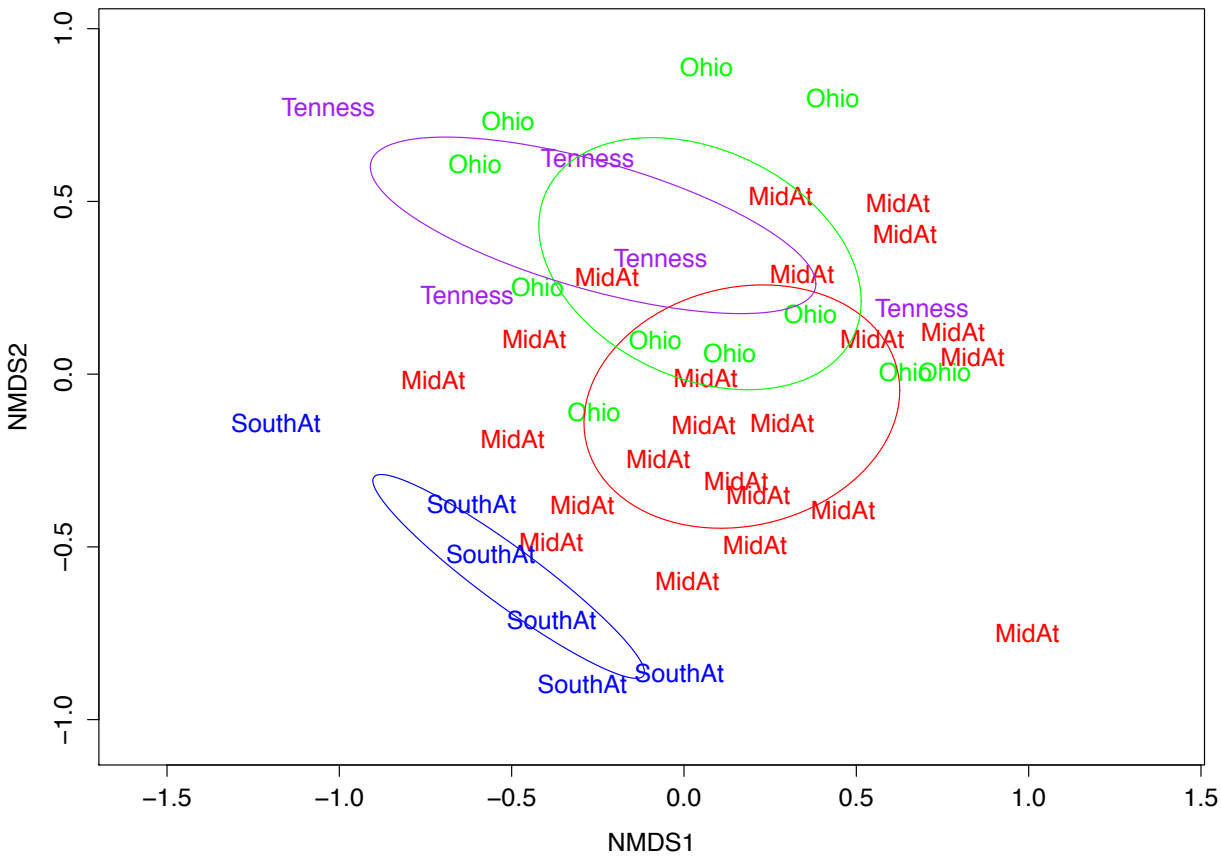


It is favorable to use non-metric multidimensional scaling (NMDS) to assist in the interpretation of ANOSIM results because both procedures are based on ranked similarities. Let's conduct a NMDS by typing:

```
spe.nmds<-metaMDS(speabu, distance='bray', k=2, autotransform=FALSE,
trymax=100)
```

Revisit the R handout for NMDS to explore the results again. Let's just plot the resulting ordination and add confidence ellipses around the sites contained in the same river basins. Type:

```
plot(spe.nmds,type='n')
text(spe.nmds,labels=sitegroup$BASIN,col=as.vector(sitegroup$COLOR))
ordiellipse(spe.nmds,sitegroup$BASIN,show.groups="MidAt",col=c("red"))
ordiellipse(spe.nmds,sitegroup$BASIN,show.groups="SouthAt",col=c("blue"))
ordiellipse(spe.nmds,sitegroup$BASIN,show.groups="Ohio",col=c("green"))
ordiellipse(spe.nmds,sitegroup$BASIN,show.groups="Tenness",col=c("purple"))
```



Next, let's perform a couple pairwise ANOSIMs to explore the differences responsible for the global ANOSIM significance reported earlier. First we will combine the species abundance and group datasets, then subset the data and perform an ANOSIM. Type:

```
data<-cbind(speabu,sitegroup)
newdata<-data[which(data$BASIN=='SouthAt'| data$BASIN=='MidAt'), ]
newdata$BASIN<-factor(newdata$BASIN)
speabu.dd<-vegdist(newdata[,1:36], "bray")
yy.anosim<-anosim(speabu.dd,newdata$BASIN)
summary(yy.anosim)

Call:
anosim(dat = speabu.dd, grouping = newdata$BASIN)
Dissimilarity: bray

ANOSIM statistic R: 0.3457
      Significance: 0.001
```

```
Based on  999  permutations

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.140 0.187 0.229 0.261

Dissimilarity ranks between and within classes:
        0%   25% 50%    75%  100%   N
Between 19 165.5 260 356.75 398.5 138
MidAt    1  81.0 170 268.00 398.5 253
SouthAt  6 110.0 258 330.00 375.0  15
```

We find a significant difference in the fish communities between sites in the South Atlantic versus the Mid-Atlantic. This matches the expectation based on the NMDS plot above. Now type:

```
newdata<-data[which(data$BASIN=='Ohio'| data$BASIN=='Tenness'), ]
newdata$BASIN<-factor(newdata$BASIN)
speabu.dd<-vegdist(newdata[,1:36], "bray")
yy.anosim<-anosim(speabu.dd,newdata$BASIN)
summary(yy.anosim)
```

```
Call:
anosim(dat = speabu.dd, grouping = newdata$BASIN)
Dissimilarity: bray

ANOSIM statistic R: 0.008112
      Significance: 0.44

Based on  999  permutations

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.174 0.226 0.275 0.335

Dissimilarity ranks between and within classes:
        0%  25%  50%  75% 100%  N
Between  2 32.0 57.0 97.5  117 55
Ohio     1 29.5 62.0 82.0  117 55
Tenness 14 27.0 79.5 93.5  112 10
```

Here, we find no difference in the Ohio vs. Tennessee River Basins.

Lastly, let's determine which species (variables) best discriminate each of the river basins (groups). This provides a way to add biological intpretation to the ANOSIM results presented above. To accomplish this we will the simper() function which performs pairwise comparisons of groups of objects and finds the average contributions of each species to the average overall Bray-Curtis dissimilarity. Currently, this function only operates on Bray-Curtis dissimilarity, which limits it's utility. Now type:

```
simper(speabu,sitegroup$BASIN)
```

```
cumulative contributions of most influential species:

$SouthAt_MidAt
 BLUECHUB  MTREDDAC  BLACDACE  ROSYDACE  MARGMADT  CRESSHIN  FALLFISH
MOTTSCUL  CREECHUB  TORRSUCK
```

```
0.1527585 0.2850426 0.3796401 0.4370021 0.4943166 0.5422522 0.5835657
0.6237794 0.6639302 0.7037381

$SouthAt_Ohio
 BLUECHUB  MTREDDAC  BLACDACE  MARGMADT  ROSYDACE  CRESSHIN  CREECHUB
ROCKBASS  FALLFISH  BLUEGILL
0.1575213 0.2754071 0.3667515 0.4311403 0.4836582 0.5356109 0.5857429
0.6315715 0.6706374 0.7096116

$SouthAt_Tenness
 BLUECHUB  MTREDDAC  STONROLL  CREECHUB  MARGMADT  BLACDACE  CRESSHIN
ROSYDACE  ROCKBASS  FALLFISH
0.1489449 0.2620912 0.3731236 0.4467574 0.5104650 0.5628355 0.6147077
0.6554122 0.6952037 0.7348283

$MidAt_Ohio
 BLACDACE  ROSYDACE  STONROLL  MOTTSCUL  ROCKBASS  BLUECHUB  TORRSUCK
TESSDART  LONGDACE  MTREDDAC  CREECHUB
0.1674126 0.2475361 0.3190813 0.3886467 0.4483055 0.5038224 0.5573978
0.6059896 0.6539054 0.6999589 0.7441246

$MidAt_Tenness
 STONROLL  BLACDACE  CREECHUB  MOTTSCUL  ROSYDACE  TORRSUCK  ROCKBASS
0.1481656 0.2892158 0.3678880 0.4431541 0.4994220 0.5497700 0.5992733

$Ohio_Tenness
 STONROLL  BLACDACE  CREECHUB  ROCKBASS  MOTTSCUL  LONGDACE  ROSYDACE
0.1714558 0.3338255 0.4363171 0.5365903 0.6026052 0.6541967 0.7033303
```

The function displays most important species for each pair of groups. These species contribute at least to 70% of the differences between groups. The function returns much more extensive results which can be accessed directly from the result object.

Full disclosure: Simper can be difficult to interpret. R documentation states “The method very badly confounds the mean between group differences and within group variation, and seems to single out variable species instead of distinctive species (Warton et al. 2012). Even if you make groups that are copies of each other, the method will single out species with high contribution, but these are not contributions to non-existing between-group differences but to within-group variation in species abundance.”

OPTIONAL READINGS

Clarke, K.R. 1993. Non-parametric multivariate analyses of changes in community structure. Australian Journal of Ecology 18:117-143.

Chapman, M.G. and A.J. Underwood. 1999. Ecological patterns in multivariate assemblages: information and interpretation of negative values in ANOSIM tests. Mar. Ecol. Prog. S. 180: 257-265.

Warton, D.I., Wright, T.W., Wang, Y. 2012. Distance-based multivariate analyses confound location and dispersion effects. Methods in Ecology and Evolution, 3: 89–101.

17. Permutational Multivariate Analysis of Variance
and Test of Multivariate Homogeneity of Group Dispersions

FISH 560: Applied Multivariate Statistics for Ecologists

R Packages: vegan
R Source: biostats

BACKGROUND

Another approach to comparing two or groups in multivariate space is known as permutational (or nonparametric) multivariate analysis of variance (perMANOVA or npMANOVA) (Anderson 2001). perMANOVA is a closely related procedure for the analysis and partitioning sums of squares using semi-metric and metric distance matrices. Insofar as it partitions sums of squares of a multivariate data set, it is directly analogous to MANOVA (multivariate analysis of variance). This method allows the use of any semi-metric (e.g. Bray-Curtis, aka Steinhaus, Czekanowski, and Sørensen) or metric (e.g. Euclidean) distance matrix (Anderson 2001, McArdle and Anderson 2001). Significance tests are done using F-tests based on sequential sums of squares from permutations of the raw data, and not permutations of residuals.

perMANOVA is quite flexible and can accommodate a wide variety of experimental designs, including nested and factorial designs consisting of both factors and continuous variables. If a significant difference between groups is detected using perMANOVA, then this could be due to differences in location in multivariate space, differences in spread, or a combination of the two. This leads us into the next approach.

Test of multivariate homogeneity of group dispersions can be used to test if the dispersions (variances) of one or more groups are different (Anderson 2006). One measure of multivariate dispersion (variance) for a group of samples is to calculate the average distance of group members to the group centroid or spatial median in multivariate space. To test if the dispersions among groups are different, the distances of group members to the group centroid are subject to ANOVA. This is a multivariate analogue of Levene's test for homogeneity of variances if the distances between group members and group centroids is the Euclidean distance. However, better measures of distance than the Euclidean distance are available for ecological data. These can be accommodated by reducing the distances produced using any dissimilarity coefficient to principal coordinates, which embeds them within a Euclidean space. The analysis then proceeds by calculating the Euclidean distances between group members and the group centroid on the basis of the principal coordinate axes rather than the original distances.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset and site group dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu if you plan to transform or standardize any of your datasets. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
sitegroup <- read.csv('MAHA_Groups.csv',header=TRUE, row.names=1)
```

PERMUTATIONAL MULTIVARIATE ANALYSIS OF VARIANCE (perMANOVA)

Here we will use the adonis() function in the vegan library to perform perMANOVA. Its usage is:

```
adonis(formula, data, permutations = 999, method = "bray",
       strata = NULL, contr.unordered = "contr.sum",
       contr.ordered = "contr.poly", ...)
```

Where:

- **`formula`** is a typical model formula such as Y ~ A + B*C, but where Y is either a dissimilarity object (inheriting from class "dist") or data frame or a matrix; A, B, and C may be factors or continuous variables. If a dissimilarity object is supplied, no species coefficients can be calculated
- **`data`** the data frame from which A, B, and C would be drawn
- **`permutations`** number of replicate permutations used for the hypothesis tests (*F* tests).
- **`method`** the name of any method used in vegdist to calculate pairwise distances if the left hand side of the formula was a data frame or a matrix
- **`strata`** groups (strata) within which to constrain permutations
- …

For more details and documentation type **`?anodis`**.

Let's conduct perMANOVA to test for differences in fish species composition between sites in different major river basins. Type:

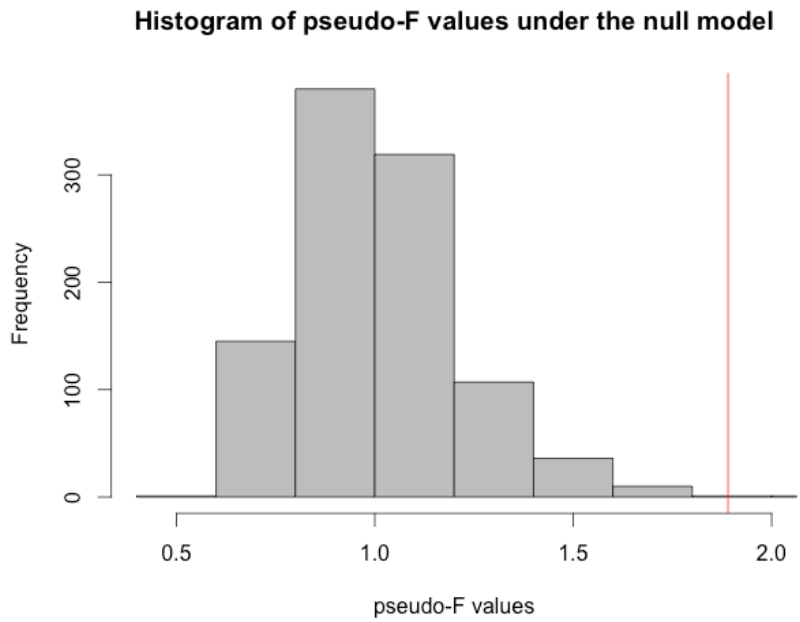
```
spe.perm<-adonis(speabu~BASIN,data=sitegroup,permutations=1000,method='bray')
spe.perm

Call:
adonis(formula = speabu ~ BASIN, data = sitegroup, permutations = 1000,
method = "bray")

Terms added sequentially (first to last)

          Df SumsOfSqs MeanSqs F.Model      R2   Pr(>F)
BASIN      3    1.9955 0.66517  1.8903 0.12151 0.003996 **
Residuals 41   14.4271 0.35188         0.87849
Total     44   16.4226                 1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results contain the typical analysis of variance table showing sources of variation, degrees of freedom, sequential sums of squares, mean squares, F statistics, partial R-squared and P values, based on N permutations.

Note the use of the formula to specify the model of the form, Y~X. In this case, X is a single four-level factor (i.e., river basins), but the right-hand side of the equation can consist of any suitable linear model, including nested and factorial designs, consisting of factors and/or continuous variables.

Let's quickly visualize the distribution of pseudo-F values under the null hypothesis to confirm the calculation of the p-value above, by typing:

```
hist(spe.perman$f.perms,main="Histogram of pseudo-F values under the null model",xlab="pseudo-F values",xlim=c(0.4,2.0),col="gray")

abline(v=1.89,col="red")
```



Note that we defined the red vertical line to be equal to the observed pseudo-F value. It is the fourth largest value; hence a p-value of 0.004 (listed above).

Now, let's test for differences in fish community composition among river basins and human disturbance according to 4 levels that are classified as low, moderate, high and very high. Try typing:

```
adonis(speabu~BASIN+DISTURB,data=sitegroup,perm=1000,method='bray')

Call:
adonis(formula = speabu ~ BASIN + DISTURB, data = sitegroup,
permutations = 1000, method = "bray")

Terms added sequentially (first to last)

          Df SumsOfSqs MeanSqs F.Model      R2   Pr(>F)
BASIN      3    1.9955 0.66517 1.87950 0.12151 0.003996 **
DISTURB    3    0.9786 0.32619 0.92166 0.05959 0.652348
Residuals 38   13.4485 0.35391         0.81890
Total     44   16.4226                 1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results present the 2-way perMANOVA comparing differences in fish community composition among river basins and human impact levels. You see that we find little evidence for fish community differences between rivers in different categories of human disturbance.

MULTIVARIATE HOMOGENEITY OF GROUPS DISPERSIONS (VARIANCES)

Next, we will test for differences in group dispersions using the betadisper() function in the vegan library. Its usage is:

```
betadisper(d, group, type = c("centroid", "median"))
```

Where:

- **d** is a distance structure such as that returned by dist, betadiver or vegdist
- **group** vector describing the group structure
- **type** indicates whether distances are calculated to the centroid of the group or the median of the group in multivariate space

For more details and documentation type **?betadisper**.

Let's start by calculating the multivariate dispersions in fish community composition between major river basins.

The first step is the selection of a distance matrix to describe similarities among objects according to the descriptors. Again, any (dis) similarity coefficient can be selected. For today, let's generate a distance matrix using an appropriate dissimilarity coefficient for species abundance – Bray-Curtis coefficient. We'll use the vegdist() function (see Ecological Resemblace chapter). Type:

```
speabu.d<-vegdist(speabu,"bray")
```

Next, let's conduct the test of multivariate dispersion by typing:

```
sp.bdp<-betadisper(speabu.d,sitegroup$BASIN)

sp.bdp
```

```
Homogeneity of multivariate dispersions

Call: betadisper(d = speabu.d, group = sitegroup$BASIN)

No. of Positive Eigenvalues: 31
No. of Negative Eigenvalues: 13

Average distance to median:
  MidAt    Ohio SouthAt Tenness
 0.5710  0.5586  0.5468  0.5271

Eigenvalues for PCoA axes:
 PCoA1  PCoA2  PCoA3  PCoA4  PCoA5  PCoA6  PCoA7  PCoA8
2.9339 1.8873 1.4983 1.2110 1.0630 0.9796 0.8902 0.8444
```

Next, let's perform the ANOVA test by typing:

```
anova(sp.bdp)
```

```
Analysis of Variance Table

Response: Distances
          Df  Sum Sq   Mean Sq F value Pr(>F)
```

```
Groups     3 0.00926 0.0030853  0.3227 0.8089
Residuals 41 0.39203 0.0095618
```

You will notice that we find no significant difference in the between-site variation in fish community composition among river basins. We can also calculate significant levels using a permutational approach by typing:

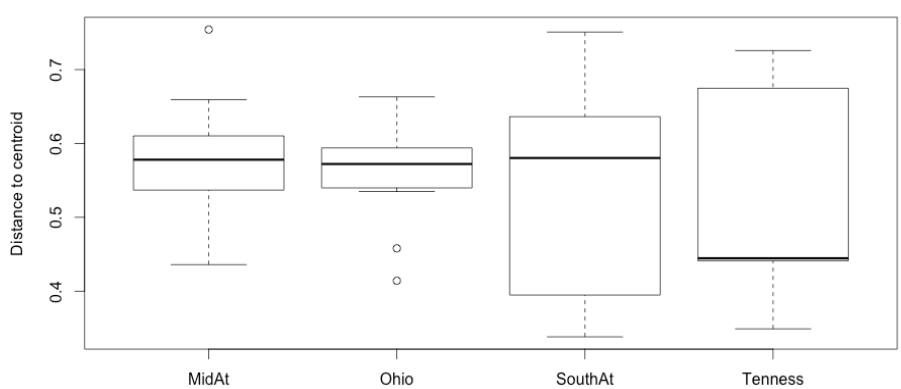
`permutest(sp.bdp,pairwise=TRUE)`

```
Permutation test for homogeneity of multivariate dispersions

Permutation Design:
Blocks:
  Defined by: none
Plots:
  Defined by: none
Within Plots:
  Permutation type: free

Permutation details:
  Number of permutations requested: 999
  Max. number of permutations allowed: 9999
  Evaluate all permutations?: No.  Activation limit: 99

Response: Distances
          Df  Sum Sq   Mean Sq      F N.Perm Pr(>F)
Groups     3 0.00926 0.0030853 0.3227    999  0.796
Residuals 41 0.39203 0.0095618

Pairwise comparisons:
(Observed p-value below diagonal, permuted p-value above diagonal)
          MidAt    Ohio SouthAt Tenness
MidAt           0.64700 0.58800   0.315
Ohio    0.64264         0.82600   0.589
SouthAt 0.57346 0.83102           0.847
Tenness 0.34222 0.59402 0.84177
```

The ANOVA and permutation test demonstrate little differences between river basins with respect to the degree of variation in fish community composition. Pairwise comparisons are also presented.

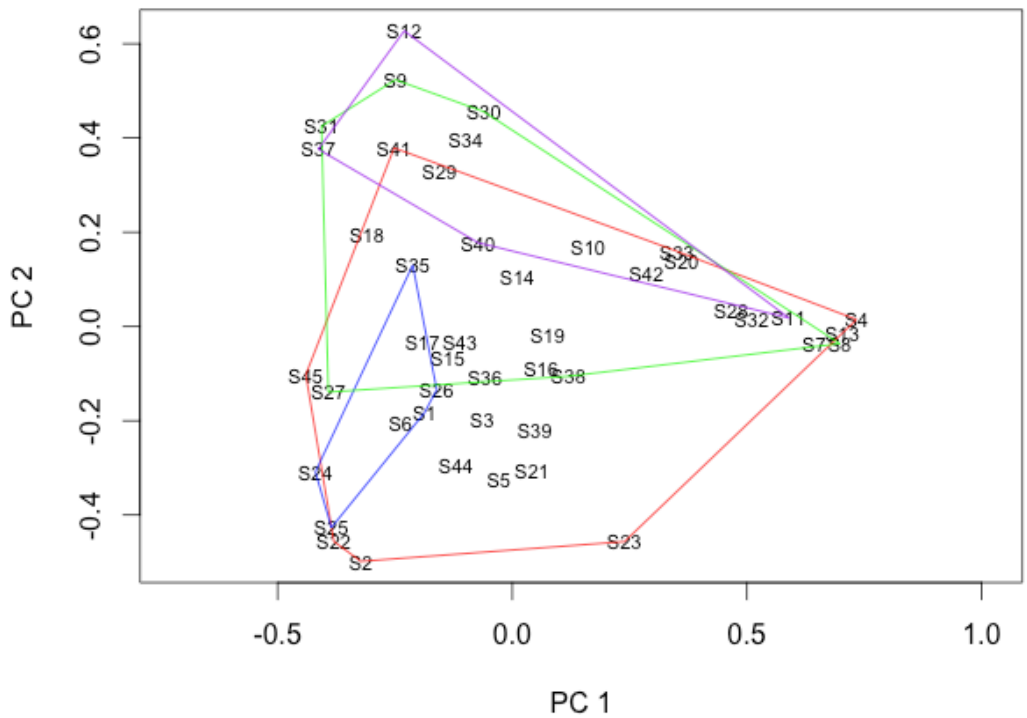
We can plot a boxplot of distances by typing:

`boxplot(sp.bdp)`



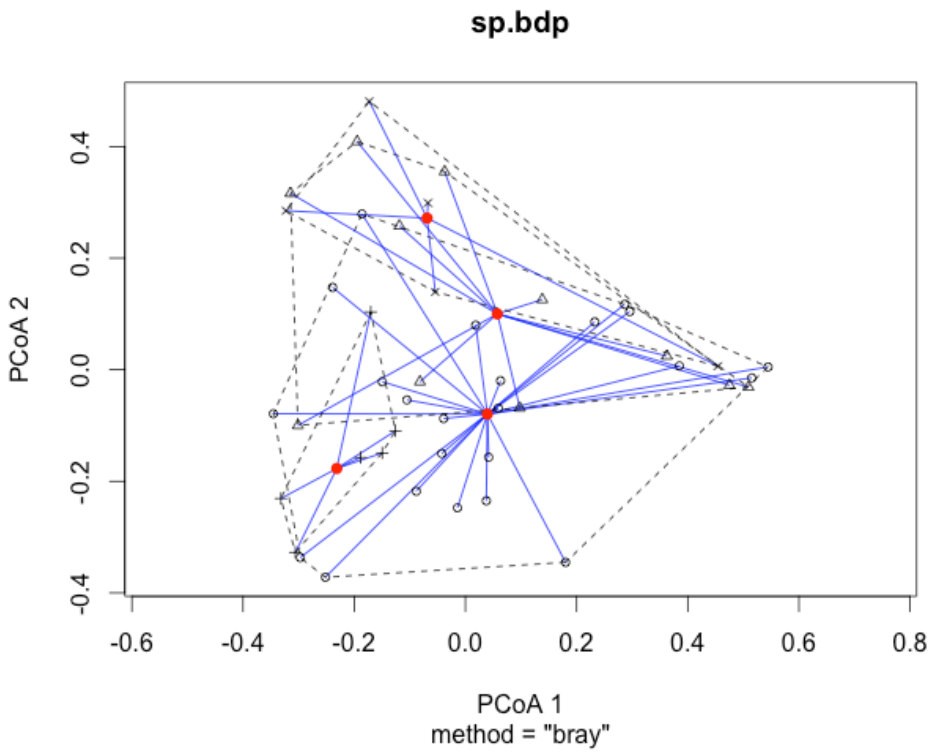
Finally, let's select an appropriate ordination approach to visual our data, namely PCoA. Let's use the code from Chapter 7 to perform PCoA.

```
spe.pcoa<-cmdscale(speabu.d, eig=TRUE, add=T)

ordiplot(spe.pcoa, choices = c(1, 2), type="text", display="sites", xlab="PC
1", ylab="PC 2")

ordihull(spe.pcoa,sitegroup$BASIN,show.groups="MidAt",col=c("red"))
ordihull(spe.pcoa,sitegroup$BASIN,show.groups="SouthAt",col=c("blue"))
ordihull(spe.pcoa,sitegroup$BASIN,show.groups="Ohio",col=c("green"))
ordihull(spe.pcoa,sitegroup$BASIN,show.groups="Tenness",col=c("purple"))
```



Alternatively, we could use the built-in graphs to visualize the data, but then make the results "presentable" by using some of the graphing techniques demonstrated in previous chapters.

```
plot(sp.bdp)
```



OPTIONAL READINGS (* recommended)

Anderson, M.J. 2001. A new method for non-parametric multivariate analysis of variance. Aust. Ecol. 26: 32–46. *

Anderson, M.J. 2001. Permutation tests for univariate or multivariate analysis of variance and regression. Can. J. Fish. Aquat. Sci. 58: 626–639.

Anderson, M.J. 2006 Distance-based tests for homogeneity of multivariate dispersions. Biometrics 62: 245–253. *

Anderson, M.J., Ellingsen, K.E. and B.H. McArdle. 2006. Multivariate dispersion as a measure of beta diversity. Ecology Letters 9: 683–693.

Anderson, M.J. and C.J.F. ter Braak 2003. Permutation tests for multi-factorial analysis of variance. J. Stat. Comput. Simul. 73: 85–113 .

McArdle, B.H. and M.J. Anderson. 2001. Fitting multivariate models to community data: a comment on distance-based redundancy analysis. Ecology 82: 290-297.

18. Mantel Test

FISH 560: Applied Multivariate Statistics for Ecologists

R Packages: vegan, fields
R Source: biostats

BACKGROUND

A common question of ecologists is whether similarities among objects described by one set of variables correspond to similarities among objects based on another set of variables. For example, is there agreement between a species composition data set and environmental dataset? The Mantel test provides a method for testing for relationships between data sets by comparing two matrices that were calculated for the same objects but that are based on two independent data sets (e.g. a species dissimilarity matrix and an environmental dissimilarity matrix for the same samples) (Mantel 1967). It calculates the correlation coefficient between corresponding positions in the two matrices, and assesses its significance based on permutations of the objects in one of the matrices.

The Mantel test is an incredibly flexible framework for conducting a wide variety of tests. For example, it may be used to assess how well observational data match an *a priori* model. In this case, the test compares the empirical resemblance matrix to a model matrix, so-called because it is constructed to represent the model to be tested. In other words, it depicts the alternative hypothesis of the test. If the model is a classification of the objects into groups, the Mantel test is then a type of non-parametric multivariate analysis of variance, and addresses a null hypothesis similar to MRPP and ANOSIM; i.e., that there are no differences between two (or more) groups defined a priori.

Alternatively, the model may involve partial dissimilarities derived by removing the effects of one or more covariate dissimilarity matrices, i.e., partial Mantel test (Smouse et al. 1986). For example, one could use a partial Mantel test to remove the effects of spatial autocorrelation. First, a Mantel test is performed between, say, an environmental distance matrix and a second matrix describing geographic distance between sites. The residuals are then submitted to a second Mantel where the residuals and site similarity based on species abundances are tested for correlation. Essentially, the environmental variables are cleansed of spatial autocorrelation. This approach provides a means of partitioning variance in a response matrix between two predictor matrixes. For more details see Smouse et al. (1986) and Legendre and Legendre (1998).

A final note that the Mantel test is also used to compute Mantel correlograms, which are often found in biogeographical studies (e.g. Mantel correlograms are usually used to detect spatial structure in species assemblages based on grouping of the response data into specific spatial distance classes). Mantel tests are then applied to each group in order to detect significant correlations at a given scale, i.e. the scales at which the data are autocorrelated (Legendre and Legendre 1998).

SET-UP

In this exercise you will be working with the MAHA species presence-absence dataset, environmental dataset and geographic dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan and fields library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Remember to install both Packages, including:

```
install.packages("fields")
library("fields")
```

Let's import the data, by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
geodata <- read.csv('MAHA_geography.csv',header=TRUE, row.names=1)
```

Next, let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

MANTEL TEST

Calculate dissimilar matrix
The first step in a Mantel test is the selection of a distance matrix to describe similarities among objects according to the descriptors. Again, any (dis) similarity coefficient can be selected. For today, let's generate a distance matrix using an appropriate dissimilarity coefficient for species occurrence – Jaccard's coefficient. We'll use the vegdist() function (see Ecological Resemblace handout). Type,

```
speocc.d<-vegdist(speocc, "jac")
```

Next, let's calculate a distance matrix based on environmental similarity using Euclidean distance:

```
envdata.d<-vegdist(envdata.tran, "euc")
```

And finally, let's calculate the geographic distances between the sample sites. There are numerous functions that perform this action, and in the simplest case we could use vegdist() using Euclidean distance and just ignore the curvature of the earth. However, here we will use a function in the fields library (remember to load this library!). Type:

```
geodata.d<-rdist.earth(geodata,miles=F)
```

Now, to first perform a Mantel test we'll use the vegan function **`mantel`**. Its usage is:

mantel(xdis, ydis, method="pearson")

Where:
- **xdis** is the explanatory distance matrix
- **ydis** is the response distance matrix
- **method** is the correlation to use; default is Pearson's correlation coefficient but Spearman rank is also available.

For additional documentation on mantel or partial mantel type **?mantel or ?mantel.partial**

Let's perform a Mantel test using the Jaccard distance matrix (species occurrence) and the Euclidean distance matrix (environment).

```
se.man<-mantel(speocc.d,envdata.d,method="pearson")
se.man
```

```
Mantel statistic based on Pearson's product-moment correlation
```

```
Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = speocc.d, ydis = envdata.d)

Mantel statistic r: 0.2225
      Significance: 0.002

Empirical upper confidence limits of r:
   90%    95%  97.5%    99%
0.0638 0.0856 0.1092 0.1355

Based on 999 permutations
```

If you get an error, then it may because there are competing versions of `mantel` in vegan and ecodist packages (you still might have ecodist loaded). In this case, type the following:

```
detach("package:ecodist", unload=TRUE)
```

Okay, from the above output we can see that the relationship is significant, which indicates that there is strong concordance between the species and environmental matrices.

Alternatively, we may want to measure the relationship between the two distance matrices using a Spearman's rank correlation coefficient, instead of the default Pearson's correlation coefficient. To do this can use the mantel2() function in BIOSTATS, which is simply an enhanced version of the mantel() function in the vegan library. Try typing

```
se.man2<-mantel(speocc.d,envdata.d,method="spear")
se.man2
```

```
Mantel statistic based on Spearman's rank correlation rho

Call:
mantel(xdis = speocc.d, ydis = envdata.d, method = "spear")

Mantel statistic r: 0.219
      Significance: 0.001

Empirical upper confidence limits of r:
   90%    95%  97.5%    99%
0.0726 0.0882 0.1092 0.1441

Based on 999 permutations
```

As before, the resulting object is a list containing several components, but the most important information is the value of the standardized Mantel test statistic, 'r', and the p-value derived from the Monte Carlo permutation test. Based on this information, can we reject the null hypothesis of no group differences? If so, is the difference between groups ecological significant as well based on the magnitude of 'r'?

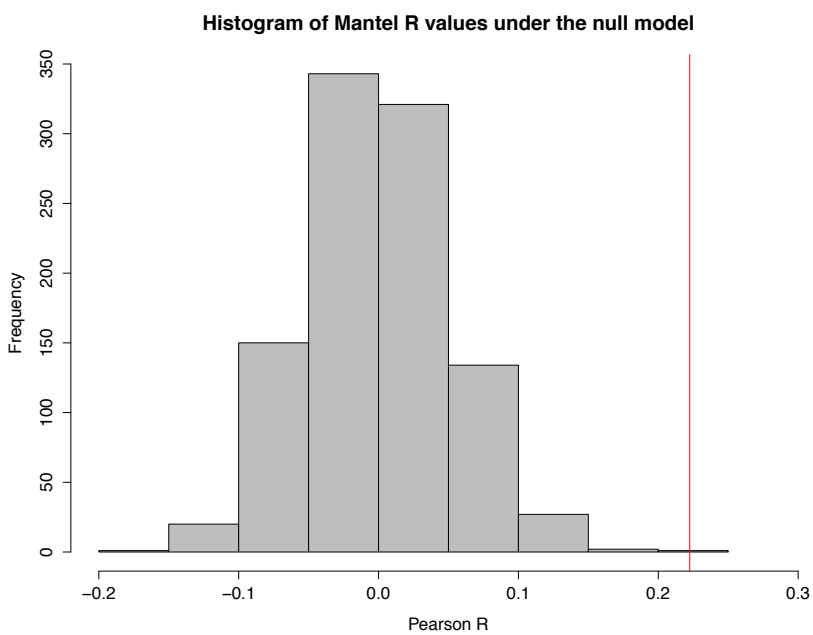
Finally, let's run the analysis between the species and geography dataset. This will test the question of whether streams located in close geographic proximity are more similar in their fish community composition compared to more distant streams.

```
se.man3<-mantel(speocc.d,geodata.d)
se.man3

Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = speocc.d, ydis = geodata.d)

Mantel statistic r: 0.3491
      Significance: 0.001

Empirical upper confidence limits of r:
   90%    95%  97.5%    99%
0.0896 0.1109 0.1357 0.1551

Based on 999 permutations
```

You will note a significant relationship between fish community composition of streams and the geographic distance between them.

We can plot the distribution of randomized Pearson R values under the null model by typing:

```
hist(se.man$perm,main="Histogram of Mantel R values under the null
model",xlab="Pearson R",xlim=c(-0.2,0.3),col="gray")
abline(v=se.man$statistic,col="red")
```



PARTIAL MANTEL TEST

Now let's test the relationship between species similarity among sites and environmental similarity after controlling for spatial differences.

Let's do this by conducting a partial Mantel test.

```
se.man4<-mantel.partial(speocc.d,envdata.d,geodata.d)
se.man4

Partial Mantel statistic based on Pearson's product-moment correlation

Call:
mantel.partial(xdis = speocc.d, ydis = envdata.d, zdis = geodata.d)

Mantel statistic r: 0.2092
      Significance: 0.001

Empirical upper confidence limits of r:
   90%    95%  97.5%    99%
0.0716 0.0915 0.1105 0.1474

Based on 999 permutations
```

OPTIONAL READINGS (* recommended)

Mantel, N. 1967. The detection of disease clustering and a generalized regression approach. Cancer Research 27:209-220.

Smouse, P.E., J.C. Long and R.R. Sokal. 1986. Multiple regression and correlation extensions of the Mantel test of matrix correspondence. Systematic Zoology 35:62 7-632.

19. Procrustes Analysis

FISH 560: Applied Multivariate Statistics for Ecologists

R Packages: vegan
R Source: biostats

BACKGROUND

An alternative approach is to assess the overall degree of association between two or more matrices applying a Procrustean superimposition approach (Gower 1971). Rather than transforming data sets into distance matrices, the raw data matrices, or their ordination solutions, are scaled and rotated in order to find an optimal superimposition that maximizes their fit. The sum of the squared residuals between configurations in their optimal superimposition can be then used as a metric of association (Gower 1971). A permutation procedure (PROTEST) implemented by Jackson (1995) can be then used to assess the statistical significance of the Procrustean fit.

The Procrustean superimposition approach provides a number of additional features, while keeping Mantel's flexibility (i.e., choice of distance measure and types of data). A particular advantage of the Procrustean approach is that when displaying multivariate data sets in their optimal superimposition, one can visually examine the concordance of observations for each dimension separately, aiding the interpretation of data structure. For the Mantel approach, since each difference between observations for all dimensions is compressed into a single distance, the importance of each variable (or dimension) cannot be evaluated directly by a simple scatterplot of one distance matrix against the other.

Although fairly common in the realm of morphometric analysis (see Rohlf and Slice 1990; Bookstein 1996 for reviews), the use of Procrustes methods in the interpretation of data sets having different characteristics (e.g., morphological vs ecological features) is restricted mainly because the possibilities of application have not been yet detailed and explored (but see Olden et al. 2001).

SET-UP

In this exercise you will be working with the MAHA species abundance dataset, environmental dataset and geographic dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the vegan library. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speocc <- read.csv('MAHA_speciesocc.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
```

Next, let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

PROCRUSTES ANALYSIS

Let's perform a Procrustes analysis. First, we'll perform NMDS on the species occurrence (Jaccard's) and the environment (Euclidean). We'll use the scores from these ordinations in the Procrustes analysis.

To perform Procrustes analysis we'll use the vegan function **`protest`** which calls on the function **`procrustes`**. Its usage is:

```
procrustes(X, Y, scale = TRUE, symmetric = FALSE, scores = "sites", ...)
```

Where:

- **`X`** is the target (explanatory) matrix (usually PCs from an ordination of the raw data)
- **`y`** is the response matrix to be rotated (usually PCs from an ordination of the raw data)
- `scale` Allow scaling of axes of Y.
- `symmetric` Use symmetric Procrustes statistic (the rotation will still be non-symmetric).
- `scores` Kind of scores used. This is the display argument used with the corresponding scores function: see scores, scores.cca and scores.cca for alternatives.
- `kind` For plot function, the kind of plot produced: kind = 1 plots shifts in two configurations, kind = 0 draws a corresponding empty plot, and kind = 2 plots an impulse diagram of residuals.
- `choices` Axes (dimensions) plotted.
- `xlab, ylab` Axis labels, if defaults unacceptable.
- `display` Show only the "target" or "rotated" matrix as points.
- `type` Combine target and rotated points with line segments or arrows.
- `permutations` Number of permutation to assess the significance of the symmetric Procrustes statistic.

For additional documentation protest type **`?procrustes`**.

First, we'll submit our species and environmental distance matrices to NMDS (refer to previous chapter).

```
spe.nmds<-metaMDS(speocc, distance='jac', k=3, autotransform=FALSE,
trymax=100)

env.nmds<-metaMDS(envdata.tran, distance='euc', k=3, autotransform=FALSE,
trymax=100)
```

Next, we will perform the Procrustes analysis, by typing:

```
sp.pro<-procrustes(spe.nmds, env.nmds)
```

Then, we will perform the randomization test (PROTEST) to determine the statistical significance of the between-matrix association, by typing:

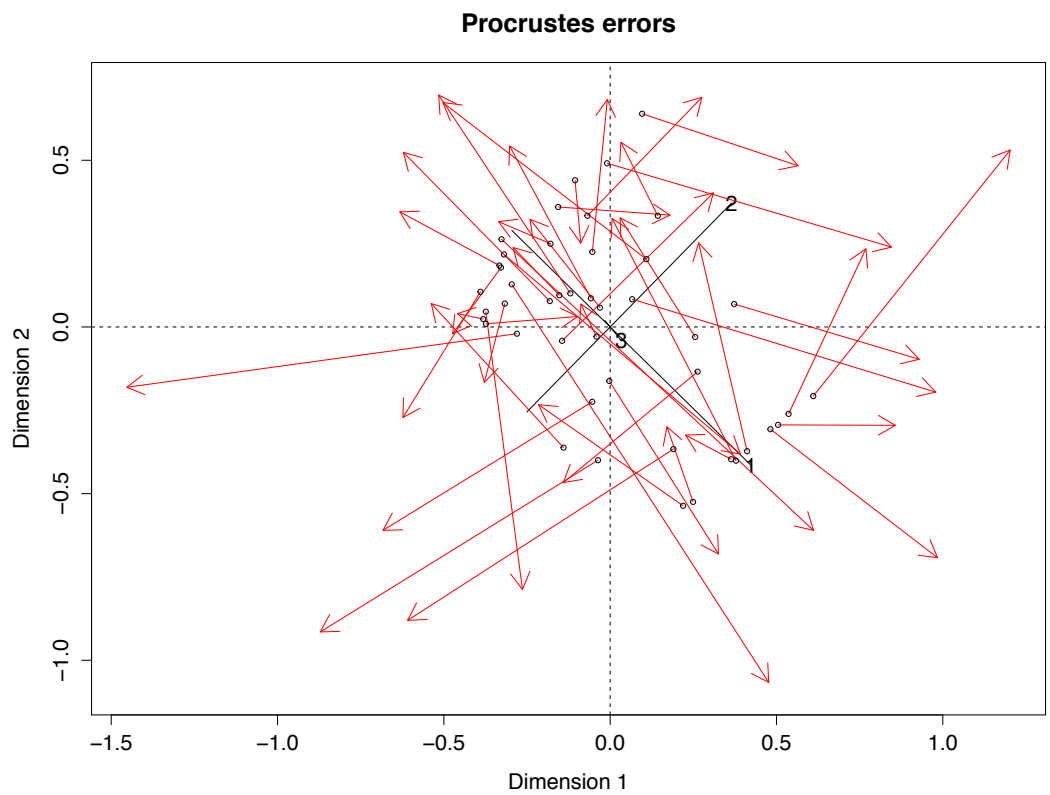
```
protest(spe.nmds, env.nmds)
```

```
Call:
protest(X = spe.nmds, Y = env.nmds)

Procrustes Sum of Squares (m12 squared): 0.7673
```

```
Correlation in a symmetric Procrustes rotation: 0.4824
Significance:  0.001
Based on 999 permutations.
```

From these results we can see that the relationship between the two datasets is significant at p=0.001. Now, lets visualize the scaled and rotated ordination superimposition resulting from the Procrustes analysis, by typing:

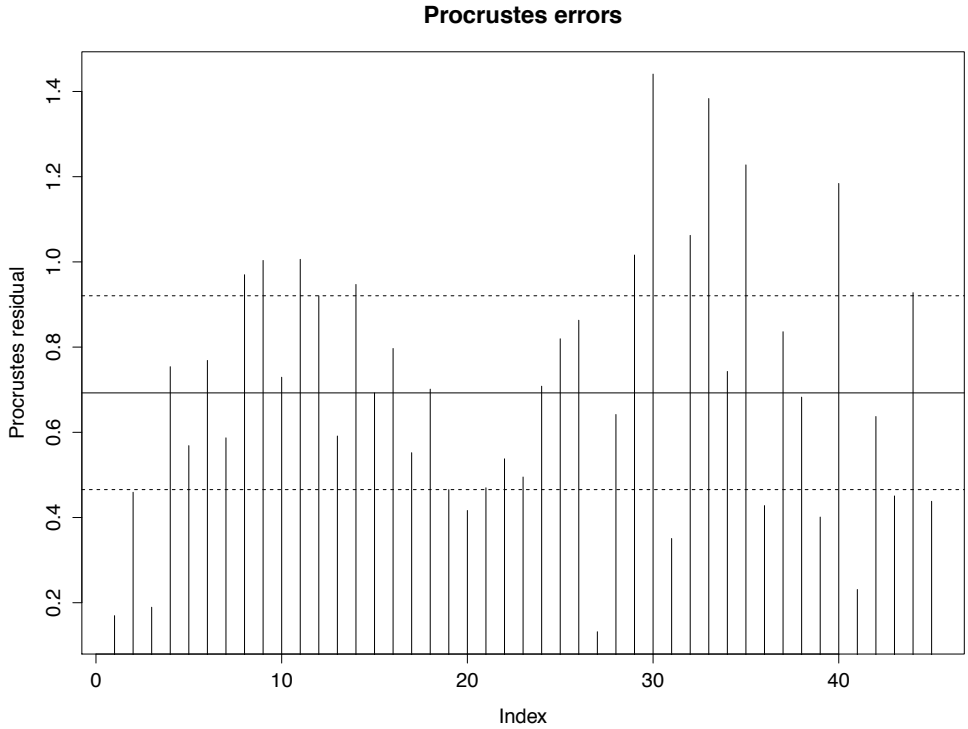
`plot(sp.pro,kind=1,ar.col="red",len=0.2)`



Again, an advantage of Procrustes analysis over the Mantel test is the ability to inspect which objects (e.g., sites) exhibit good or poor matches with their corresponding environmental data. On the left, the blue arrows indicate the displacement of objects (here, sites) between the x matrix (environment) and the y matrix (species abundances). The longer the arrow, the larger the mismatch between environmental similarity and species similarity.

We can also examine a bar graph of the residuals by typing

`plot(sp.pro,kind=2)`



For this example we used the first three axes resulting from NMDS. Analytically, there is no limit to the number of PCs that one chooses to include in a Procrustes rotation, but visual aids are unavailable when more than three dimensions are included.

PARTIAL PROCRUSTES TEST

Now let's test the relationship between species similarity among sites and environmental similarity after controlling for spatial differences. Let's do this by conducting a partial Procrustes test.

First, we factor out environmental differences among sites that are explained by differences in space (geographic coordinates). Notice how we use the raw geographic coordinate matrix since we don't need to conduct a NMDS on a matrix with just 2 variables, and we recalculated the environmental NMDS using just 2 dimensions to match.

```
env.nmds<-metaMDS(envdata.tran, distance='euc', k=2, autotransform=FALSE,
trymax=100)

sp.pro<-protest(env.nmds, geodata)
```

Next we determine the residual variation by calculating the residuals between the position of the sites in environmental versus geographic ordination space (X is based on nmdsX and Yrot is the rotated positions of nmdsY).

```
res<-sqrt((sp.pro$X-sp.pro$Yrot)^2)
```

Type res to examine the residuals.

Finally, now that you have the residuals, we can calculate the relationship between fish community differences (2-d solution in NMDS) among the sites and environmental differences among the sites (after we accounted for geographic differences).

```
spe.nmds<-metaMDS(speocc, distance='jac', k=2, autotransform=FALSE,
trymax=100)

sp.pro<-protest(spe.nmds,res)
sp.pro

Call:
protest(X = spe.nmds, Y = res)

Procrustes Sum of Squares (m12 squared): 0.826
Correlation in a symmetric Procrustes rotation: 0.4171
Significance:  0.001

Permutation: free
Number of permutations: 999
```

OPTIONAL READINGS (* recommended)

Bookstein FL (1996) Biometrics, biomathematics and the morphometric synthesis. Bull Math Biol 58:313–365

Gower JC (1971) Statistical methods of comparing different multivariate analyses of the same data. In: Hodson FR, Kendall DG, Tautu P (eds) Mathematics in the archaeological and historical sciences. Edinburgh University Press, Edinburgh, pp 138–149

Jackson DA (1995) PROTEST: a Procrustean randomization test of community environment concordance. Écoscience 2:297–303

Olden JD, Jackson DA, Peres-Neto PR (2001) Spatial isolation and fish communities in drainage lakes. Oecologia 127:572–585

Peres-Neto, P.R. and D.A. Jackson (2001) How well do multivariate datasets match? The advantages of a Procrustean superimposition approach over the Mantel test. Oecologia 129: 169-178 *

Rohlf FJ, Slice DE (1990) Extensions of the Procrustes method for the optimal superimposition of landmarks. Syst Zool 39:40–59

20. RLQ and Fourth Corner Analysis

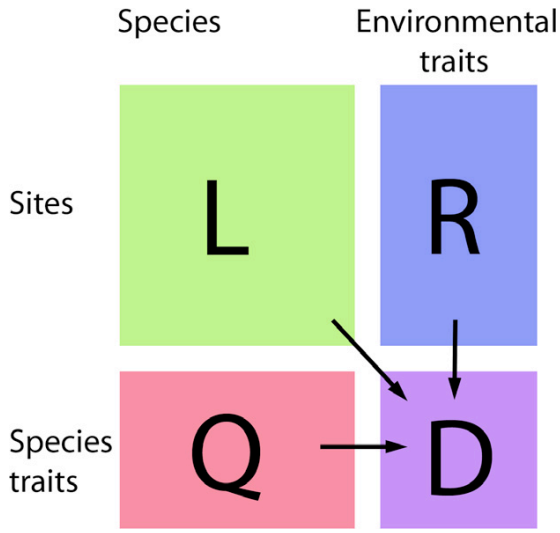
FISH 560: Applied Multivariate Statistics for Ecologists

Topics

- RLQ Analysis
- Fourth-corner Analysis

R Packages: ade4, vegan, dummies
R Source: biostats

BACKGROUND



The challenge of associating species traits and environmental variables using species abundance or presence/absence data is known as the fourth-corner problem in ecology (Legendre et al. 1997). The fourth-corner problem can be thought of as a three table problem, which takes matrices of environmental data (R), species abundance or presence/absence data (L) and species trait data (Q), and uses these three tables to infer how species traits relate to the environment (D) (Brown et al. 2014). Two strategies for the fourth-corner problem have been proposed.

RLQ analysis is an exploratory ordination approach to the fourth-corner problem (Doledec et al. 1996). Pairs of ordinations are jointly constructed via a generalized singular value decomposition, which can be understood as jointly relating each of the environmental variables and species traits to species abundance or presence/absence. This can provide a broad qualitative overview of how traits and environmental variables are associated, as opposed to specific details.

A hypothesis testing approach to the fourth-corner problem has also been proposed (Legendre et al. 1997). In this approach, the fourth-corner problem is considered as a matrix algebra problem, and R, L and Q are used to determine a matrix D (the matrix product of R'LQ) which summarizes the association between environmental variables and species traits. Permutation tests are then used to test hypotheses about the environment-trait association. Let's explore both of these approaches in this tutorial.

SET-UP

In this exercise you will be working with the MAHA species abundance dataset, environment dataset, and species trait dataset. But first remember to set-up your R work session by defining the current work directory to your folder of choice and loading the appropriate libraries. Also, make sure to source the BIOSTATS file from the *File* pull-down menu. You can also do this using the functions **`setwd`**, **`library`** and **`source`**.

Let's import the data, by typing:

```
speabu <- read.csv('MAHA_speciesabu.csv',header=TRUE, row.names=1)
envdata <- read.csv('MAHA_environment.csv',header=TRUE, row.names=1)
spetra <- read.csv('MAHA_speciestrait.csv',header=TRUE, row.names=1)
```

Let's take a quick look at the species dataset to remind ourselves of the data structure.

```
str(spetra)
```

```
'data.frame':     36 obs. of  8 variables:
 $ MaxLen  : int  78 180 100 410 260 100 860 180 300 360 ...
 $ TempPref: Factor w/ 5 levels "Cold","Cold/cool",..: 5 4 3 5 3 4 1 4 3 3
 $ SubPref : Factor w/ 4 levels "Rubble","Sand",..: 1 1 1 4 1 4 1 1 1 2 ...
 $ VertPos : Factor w/ 2 levels "Benthic","Non-benthic": 1 1 1 2 1 2 2 2 2 2
 $ Trophic : Factor w/ 5 levels "Herbivore-detritivore",..: 2 2 4 3 4 4 3 4 3
 $ Long    : int  4 6 4 5 11 6 15 7 9 7 ...
 $ Maturity: num  1.5 2 2 2.5 1.5 1.2 2.5 2 2 1.5 ...
 $ Fecund  : int  156 692 4758 40950 755 5382 3456 2170 4327 45257 ...
```

You will see that each species (n=36) has information associated with 8 traits. Some traits are quantatitive (MaxLen: maximum length; Long: longevity) and others are categorical (TempPref: temperature preference; Trophic: trophic guild).

Now we need to transform the various datasets before diving into the analysis.

First, let's recode the trait matrix so that the categorical traits (currently text) are presented as dummy variables (i.e., 0 and 1 to indicate the absence or presence of each trait state).

```
spetra.tran <- dummy.data.frame(spetra)
```

Type **`str(spetra.tran)`** to take a quick look. Next, let's standardize the traits by re-scaling by minimum-to-maximum following the recommendation of McCune (2015).

```
spetra.tran <- data.stand(spetra.tran,method='range',margin='column')
```

Second, let's conduct a 'row normalization' (i.e., to rescale each row so that the sums equal 1, or in other words to calculate relative species abundances), type the following:

```
speabu.tran<-data.stand(speabu,method='total',margin='row',plot=F)
```

Finally, let's log (based 10) transform the environmental dataset.

```
envdata.tran<-data.trans(envdata,method='log',plot=F)
```

CONDUCTING RLQ ANALYSIS

A preliminary step of RLQ analysis is to perform the separate analyses of each data matrix. In this tutorial we will use the rlq() function in the ade4 library. Its usage is:

```
rlq (dudiR, dudiL, dudiQ, …)
```

Where a selection of the arguments include:

`dudiR`	a duality diagram providing from one of the functions dudi.hillsmith, dudi.pca, ...
`dudiL`	a duality diagram of the function dudi.coa
`dudiQ`	a duality diagram providing from one of the functions dudi.hillsmith, dudi.pca, ...
`scannf`	a logical value indicating whether the eigenvalues bar plot should be displayed

…

For additional documentation type **?rlq**.

Let's perform the RLQ analysis.

First, correspondence analysis is applied to the species relative abundance matrix by typing:

```
L.species <- dudi.coa(speabu.tran, scannf = FALSE)
```

Second, principal component analysis is applied on the quantitative trait matrix by typing:

```
Q.trait <- dudi.pca(spetra.tran, row.w = L.species$cw, scannf = FALSE)
```

Third, principal component analysis is applied on the environmental matrix by typing:

```
R.env <- dudi.pca(envdata.tran, row.w = L.species$lw,scannf = FALSE)
```

Lastly, to complete the RLQ analysis, separate analyses of traits and environmental variables should be weighted by the sites and species weights derived from the previous correspondence analysis.

```
rlq.MAHA <- rlq(R.env, L.species, Q.trait, scannf = FALSE)
```

RLQ analysis finds coefficents (in $c1) to obtain a linear combination of traits (species scores in $lQ) and coefficients (in $l1) to obtain a linear combination of environmental variables (site scores in $lR). The covariance between these two sets scores is maximized and equal to the square root of the corresponding eigenvalue.

As RLQ analysis maximizes the covariance between the traits and the environmental variables mediated by the species abundances, it is important to see how the individual parts (i.e. cov(traits, env)^2 = var(traits) × var(env) × cor(traits, env)^2) of the compromise are considered. It is possible to compare the RLQ analysis to the separate analyses which maximize independently the structure of the trait (PCA of the traits), the structure of the environment (PCA of the environmental variables) and the correlation (CA of the sites-species matrix). These comparisons are provided by the summary function.

```
summary(rlq.MAHA)
```

```
RLQ analysis

Class: rlq dudi
Call: rlq(dudiR = R.env, dudiL = L.species, dudiQ = Q.trait, scannf = FALSE)

Total inertia: 3.06

Eigenvalues:
    Ax1     Ax2     Ax3     Ax4     Ax5
1.76265 0.70484 0.31385 0.13984 0.06701

Projected inertia (%):
    Ax1     Ax2     Ax3     Ax4     Ax5
  57.60   23.03   10.26    4.57    2.19

Cumulative projected inertia (%):
    Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
  57.60   80.64   90.89   95.46   97.65

(Only 5 dimensions (out of 10) are shown)
```

```
Eigenvalues decomposition:
        eig    covar      sdR      sdQ      corr
1 1.7626529 1.327649 1.679339 1.573950 0.5022898
2 0.7048441 0.839550 1.309862 1.506642 0.4254130

Inertia & coinertia R (R.env):
    inertia      max     ratio
1  2.820178 3.149831 0.8953426
12 4.535917 5.030714 0.9016449

Inertia & coinertia Q (Q.trait):
    inertia      max     ratio
1  2.477317 4.570410 0.5420339
12 4.747288 7.617092 0.6232415

Correlation L (L.species):
       corr       max     ratio
1 0.5022898 0.8707074 0.5768755
2 0.4254130 0.8024854 0.5301193
```

From this output you can see that the variation of the environmental scores is well preserved on the first two axes (90.1%), and the variation of the trait scores are moderately preserved (62.3%). Similarly, the correlations for both CA axes (species scores) are fairly high.

Finally, a multivariate test can be applied to evaluate the global significance of the traits-environment relationships. This test is based on the total inertia of the RLQ analysis. Type:

```
rlqtest.MAHA <- randtest(rlq.MAHA, modeltype = 6, nrepet = 4999)

rlqtest.MAHA
```

```
Monte-Carlo tests
Call: randtest.rlq(xtest = rlq.MAHA, nrepet = 4999, modeltype = 6)

Number of tests:   2

Adjustment method for multiple comparisons:   none
Permutation number:   4999
     Test      Obs  Std.Obs   Alter Pvalue
1 Model 2 3.060001 5.854345 greater 0.0002
2 Model 4 3.060001 1.354139 greater 0.0980
```

This shows the results of model 2 (permutation of sites, i.e. rows) and 4 (permutation of species, i.e. columns), which should be combined to obtain a test with a correct type I error (Dray et al. 2014). The result indicates marginal statistical significant.

```
rlqtest.MAHA$comb.pvalue
```

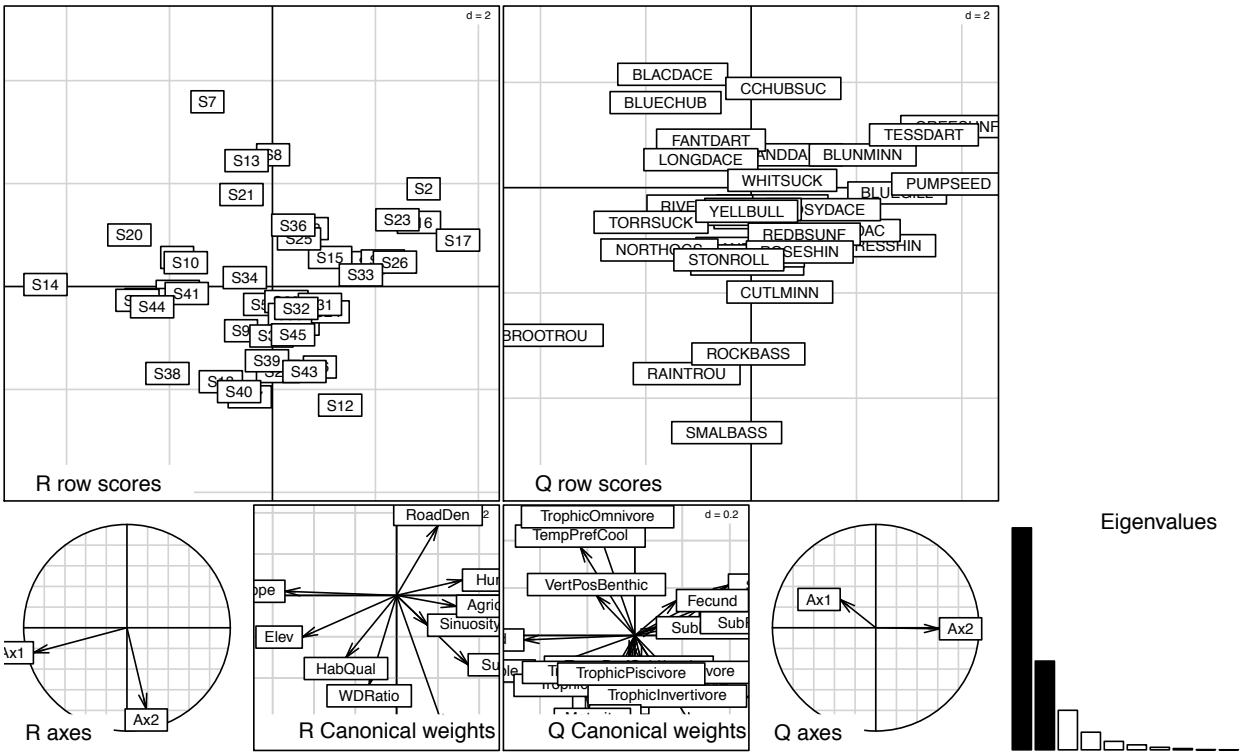
```
[1] 0.098
```

You can also plot the results from the permutation test by typing **`plot(rlqtest.MAHA)`**.

PLOTTING THE RLQ RESULTS

The different factorial maps of the analysis are obtained by the plot function:

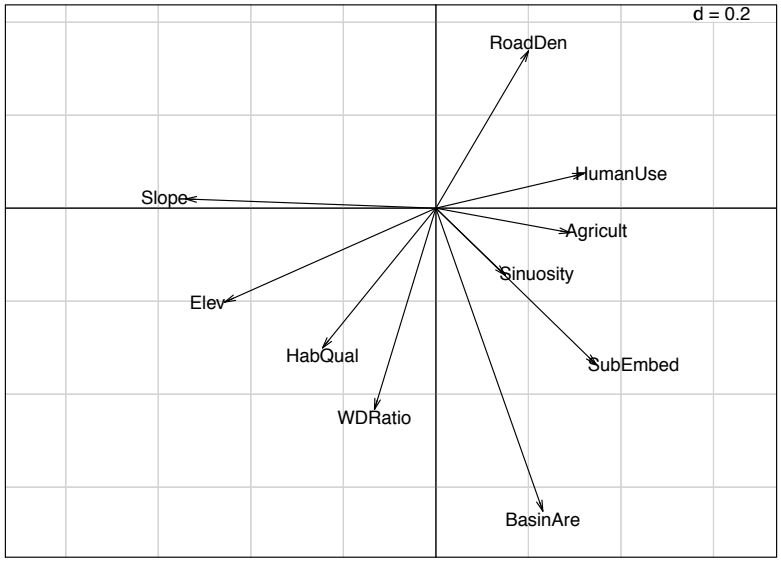
```
plot(rlq.MAHA)
```

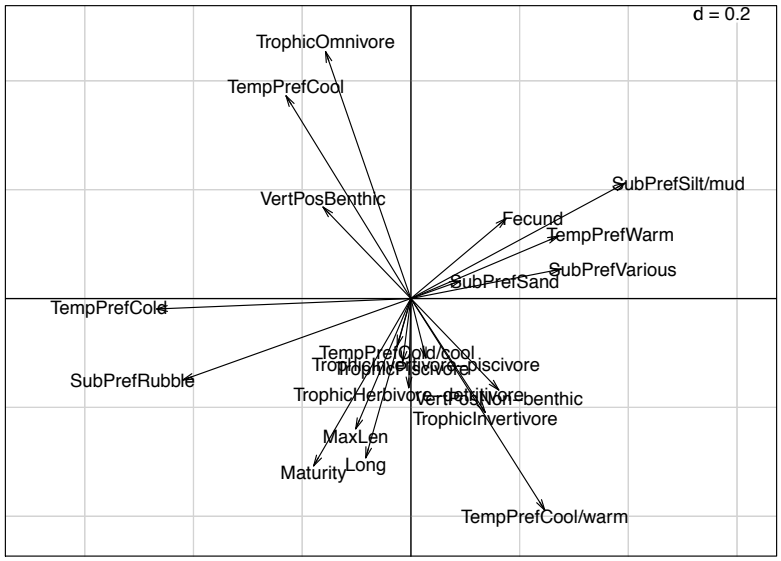


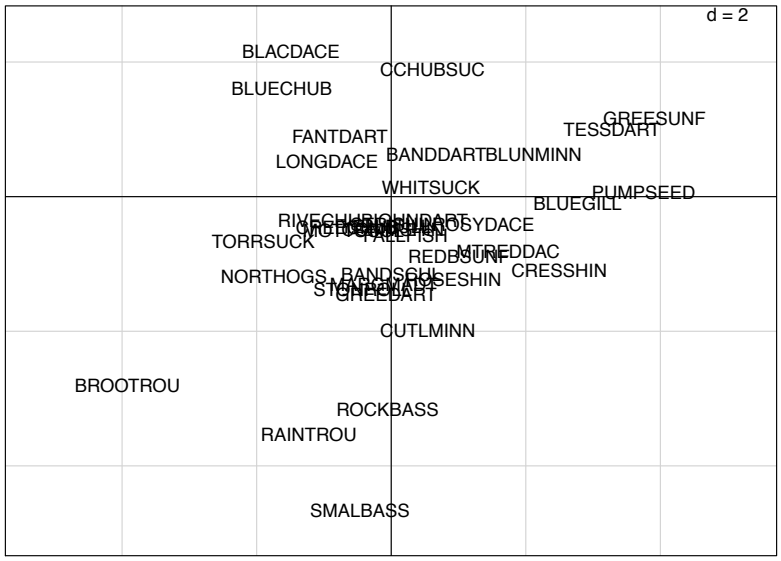
This demonstrates the correspondence analysis of the species-by-site matrix (top panels), the R axes representing the canonical weights of the environmental variables, the Q axes representing the canonical weights of the traits, and the eigenvalues (first two shown in black).

The separate factoral maps can be obtained separately by plotting the different elements contained in the rlq.MAHA object (remember that you can type **`rlq.MAHA`** to list all the available elements).

```
s.arrow(rlq.MAHA$l1, boxes = FALSE)
s.arrow(rlq.MAHA$c1, boxes = FALSE)
s.label(rlq.MAHA$lQ, boxes = FALSE)
```


d = 0.2
RoadDen
HumanUse
Slope
Agricult
Sinuosity
Elev
HabQual
SubEmbed
WDRatio
BasinAre


d = 0.2
TrophicOmnivore
TempPrefCool
SubPrefSilt/mud
VertPosBenthic
Fecund
TempPrefWarm
SubPrefVarious
SubPrefSand
TempPrefCold
SubPrefRubble
TrophicInvertivore
MaxLen
Long
Maturity
TempPrefCool/warm


d = 2
BLACDACE
CCHUBSUC
BLUECHUB
GREESUNF
TESSDART
FANTDART
BANDDARTBLUNMINN
LONGDACE
WHITSUCK
PUMPSEED
BLUEGILL
TORRSUCK
REDBSUNF
CRESSHIN
CUTLMINN
BROOTROU
ROCKBASS
RAINTROU
SMALBASS

FOURTH-CORNER ANALYSIS

Fourth-corner analysis will be accomplished using the fourthcorner() function in the ade4 library. Its usage is:

```
fourthcorner(tabR, tabL, tabQ, modeltype = 6, nrepet = 999,
p.adjust.method.G = p.adjust.methods, p.adjust.method.D =
p.adjust.methods, ...)
```

Where a selection of the arguments include:

`tabR`	a dataframe with the measurements of m environmental variables (columns) for the n sites (rows)
`tabL`	a dataframe containing the abundances of p species (columns) at n sites (rows)
`tabQ`	a dataframe describing s species traits (columns) for the p species (rows)
`modeltype`	an integer (1-6) indicating the permutation model used in testing procedure.
`nrepet`	the number of permutations
`p.adjust.method.G`	a string indicating a method for multiple adjustment used for output tabG
`p.adjust.method.D`	a string indicating a method for multiple adjustment used for output tabD/tabD2

For additional documentation type **`?fourthcorner`**.

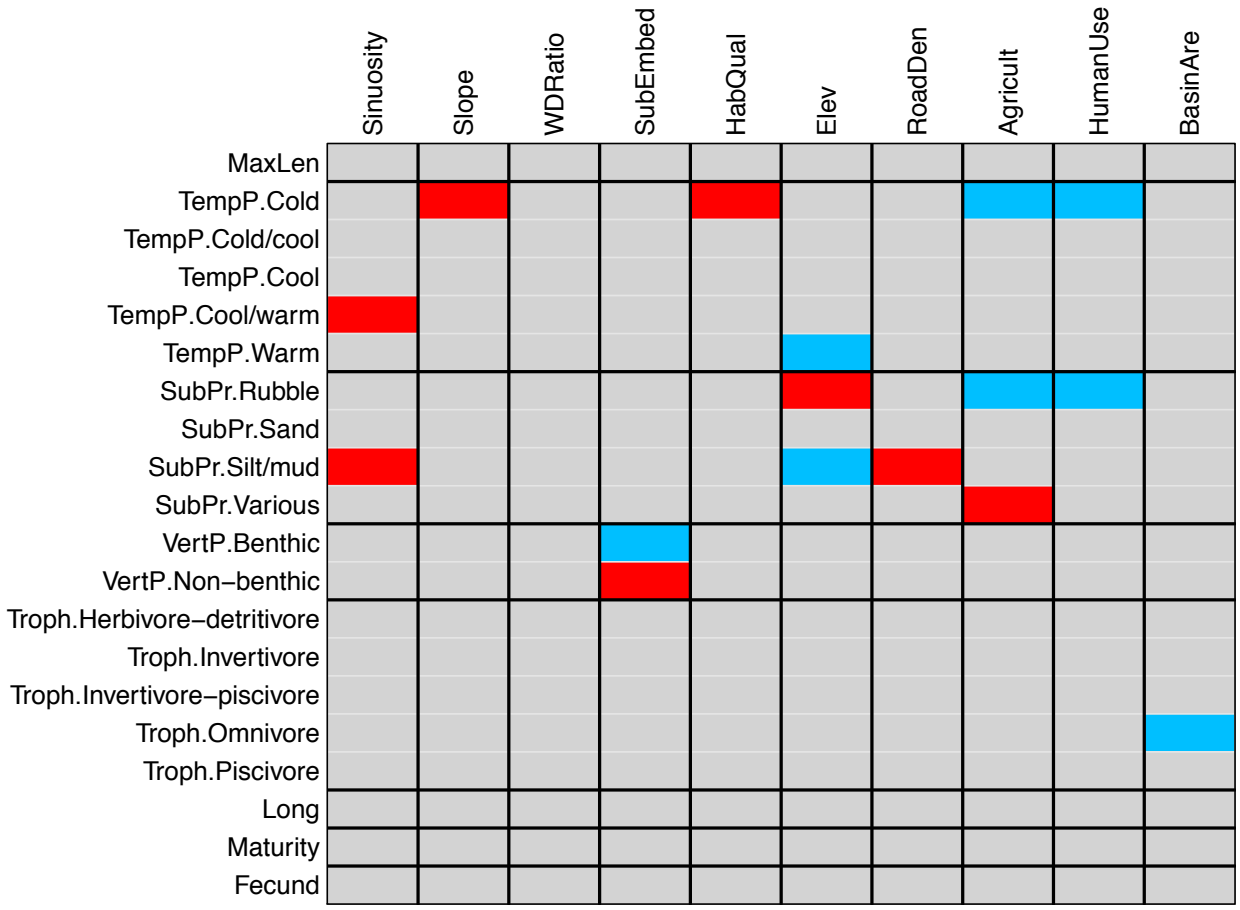
Let's perform the fourth-corner analysis. To obtain a test with a correct type I error, modeltype argument is set to 6 (following Dray et al. 2014). Note that a very high number of repetitions may be required to have enough power, but this is time-consuming and could be modified to speed up the different analyses (e.g., nrepet <- 999). Note that by default, the nrepet argument of the fourthcorner function is set to 999.

```
fourthcor.MAHA <- fourthcorner(envdata.tran, speabu.tran,spetra, modeltype =
6, p.adjust.method.D = "none", nrepet = 4999)

fourthcor.MAHA

Test   Stat                    Obs               Std.Obs     N.perm    Pvalue
1      Sinuosity / MaxLen      r -0.0321131413 -0.328924753   4999     0.723
2      Slope / MaxLen          r  0.1498299096  1.218982865   4999     0.219
3      WDRatio / MaxLen        r  0.0747526403  0.651665743   4999     0.517
4      SubEmbed / MaxLen       r  0.1284100284  0.922487431   4999     0.3658
5      HabQual / MaxLen        r  0.0856208885  0.937171998   4999     0.337
6      Elev / MaxLen           r  0.0139198589  0.135405142   4999     0.9016
7      RoadDen / MaxLen        r -0.1343229420 -1.220442196   4999     0.2388
8      Agricult / MaxLen       r -0.0691995456 -0.764822439   4999     0.4742
9      HumanUse / MaxLen       r -0.0911340316 -0.998276612   4999     0.332
10     BasinAre / MaxLen       r  0.0858872581  0.468685813   4999     0.6702
11     Sinuosity / TempP.Cold Homog.  0.00040 -0.473154610   4999     0.0928
12     Slope / TempP.Cold     Homog.  0.029415 2.928102622   4999     0.9714
…
```

This table displays the results from the pairwise comparions between each environmental variable and trait (trait state when dealing with categorical variables), including the p-value according to the permutation test.

We can also plot the results, by typing:

```
plot(fourthcor.MAHA, alpha = 0.05, stat = "D2")
```



Blue cells correspond to negative significant relationships while red cells correspond to positive significant relationships (this can be modified using the argument col). In this example, there are some associations between categorical traits and quantitative environmental variable which can be measured in three different ways (Legendre et al. 1997). These three methods correspond to three possible values of the stat argument in the plot and print functions:

- stat="D2": the association is measured between the quantitative variable and each category sepa-rately. A correlation coefficient is used to indicate the strength of the association between the given category and the small or large values of the quantitative variable.
- stat="G": the association between the quantitative variable and the whole categorical variable is measured by a global statistic (F).
- stat="D": the association is estimated between the quantitative variable and each category separately by a measure of the within-group homogeneity. The strength of the association is indicated by the dispersion of the values of the quantitative variable for a given category.

In the rest of the tutorial, we focus on the D2 statistic.

Next, we can adjust p-values for multiple comparisons (here we used the fdr method using the p.adjust.4thcorner function).

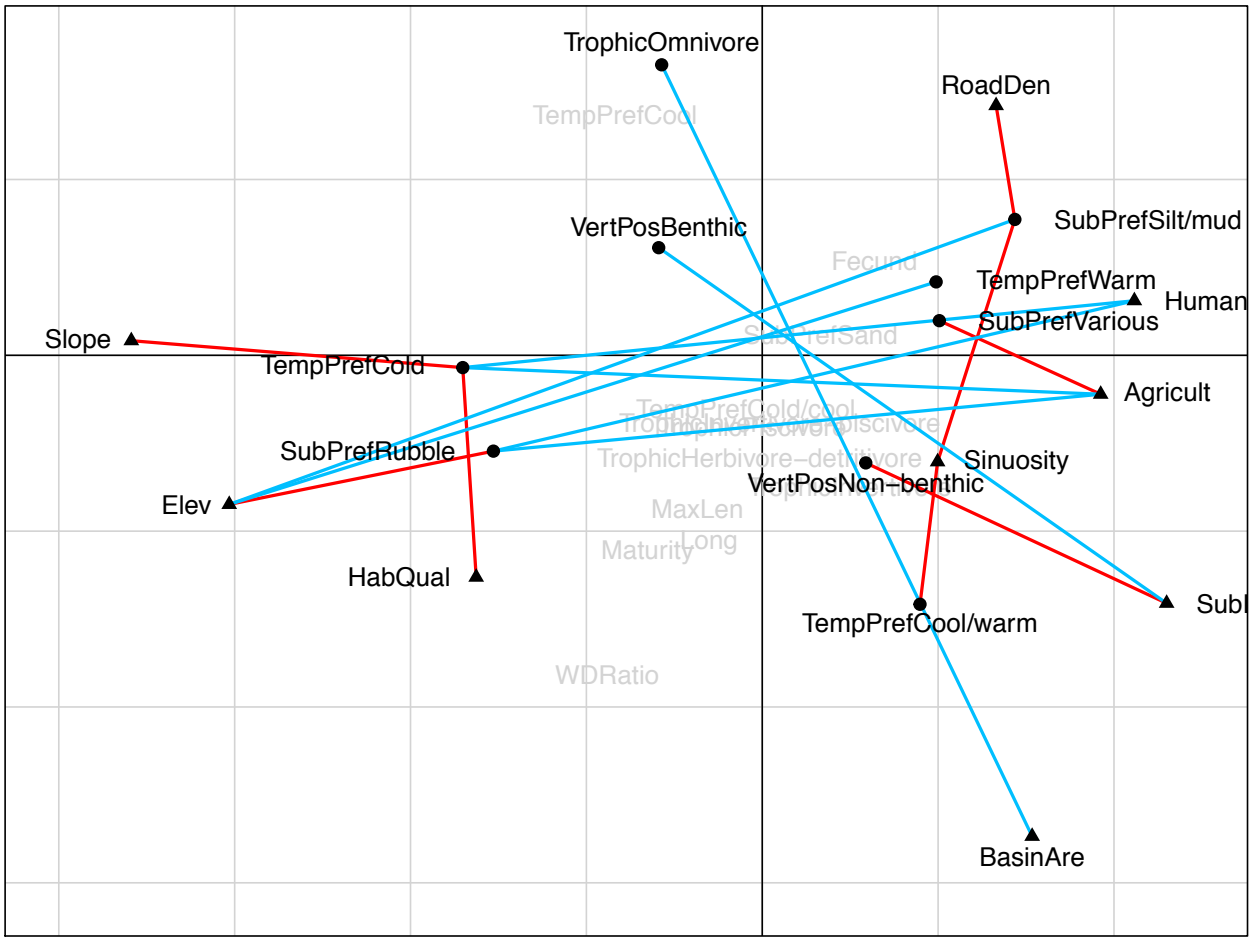
```
fourthcor2.MAHA <- fourthcorner(envdata.tran, speabu.tran,spetra, modeltype =
6, p.adjust.method.D = "fdr", nrepet = 4999)
```

Plot these results and take a look what happens!

COMBINING RLQ AND FOURTH-CORNER ANALYSIS

Both approaches can be combined if RLQ scores are used to represent traits and environmental variables on a biplot. Then, significant associations revealed by the fourth-corner approach can be represented using segments (blue lines for negative associations, red lines for positive associations, see the argument col). Only traits and environmental variables that have at least one significant association are represented on the factoral map below.

```
plot(fourthcor.MAHA,x.rlq=rlq.MAHA,alpha=0.05,stat="D2",type="biplot")
```



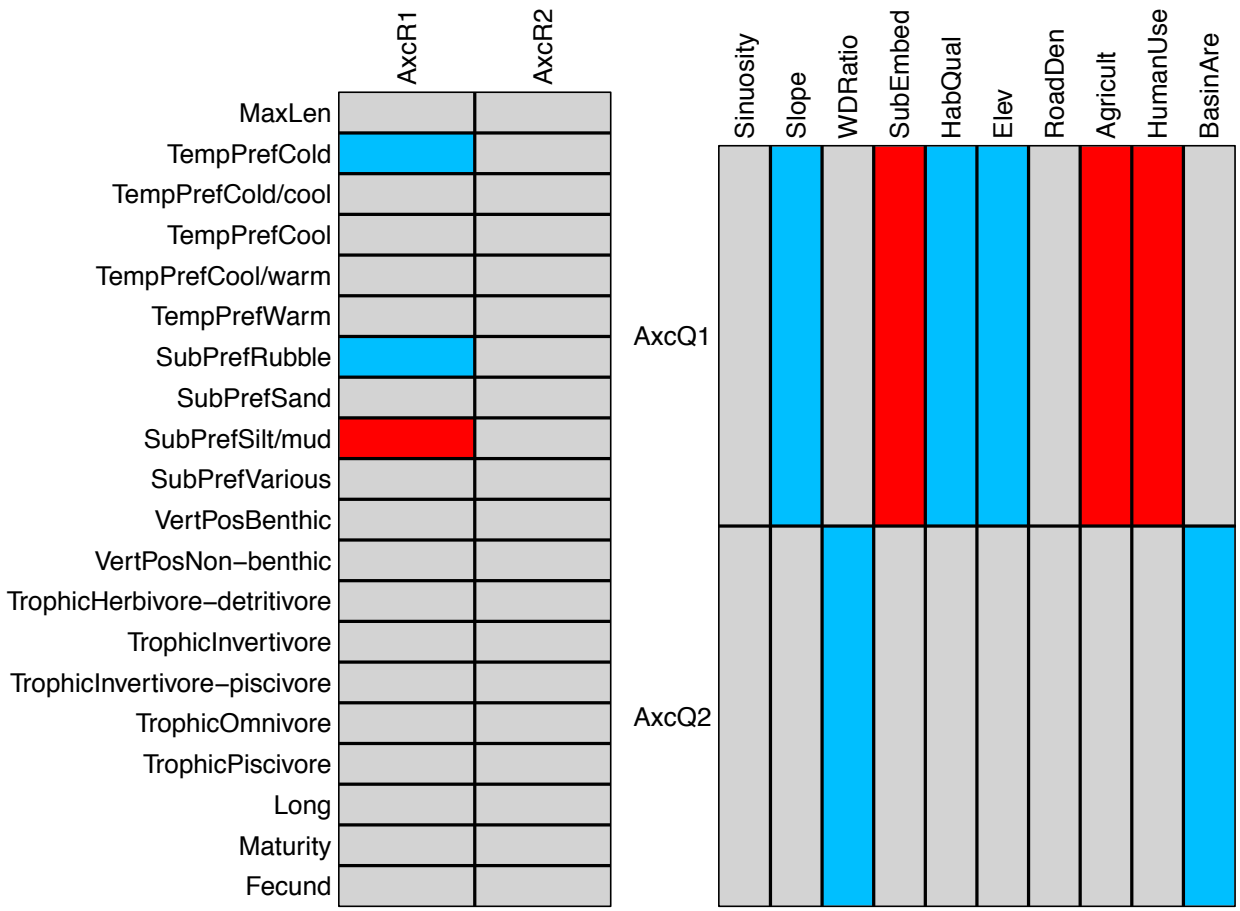
Another approach is provided by the fourthcorner.rlq function and consists in testing directly the links between RLQ axes and traits (typetest="Q.axes") or environmental variables (typetest="R.axes").

```
testQaxes<-fourthcorner.rlq(rlq.MAHA,modeltype = 6, typetest="Q.axes",
p.adjust.method.D = "fdr", nrepet = 4999)
```

```
testRaxes<-fourthcorner.rlq(rlq.MAHA,modeltype = 6, typetest="R.axes",
p.adjust.method.D = "none", nrepet = 4999)
```

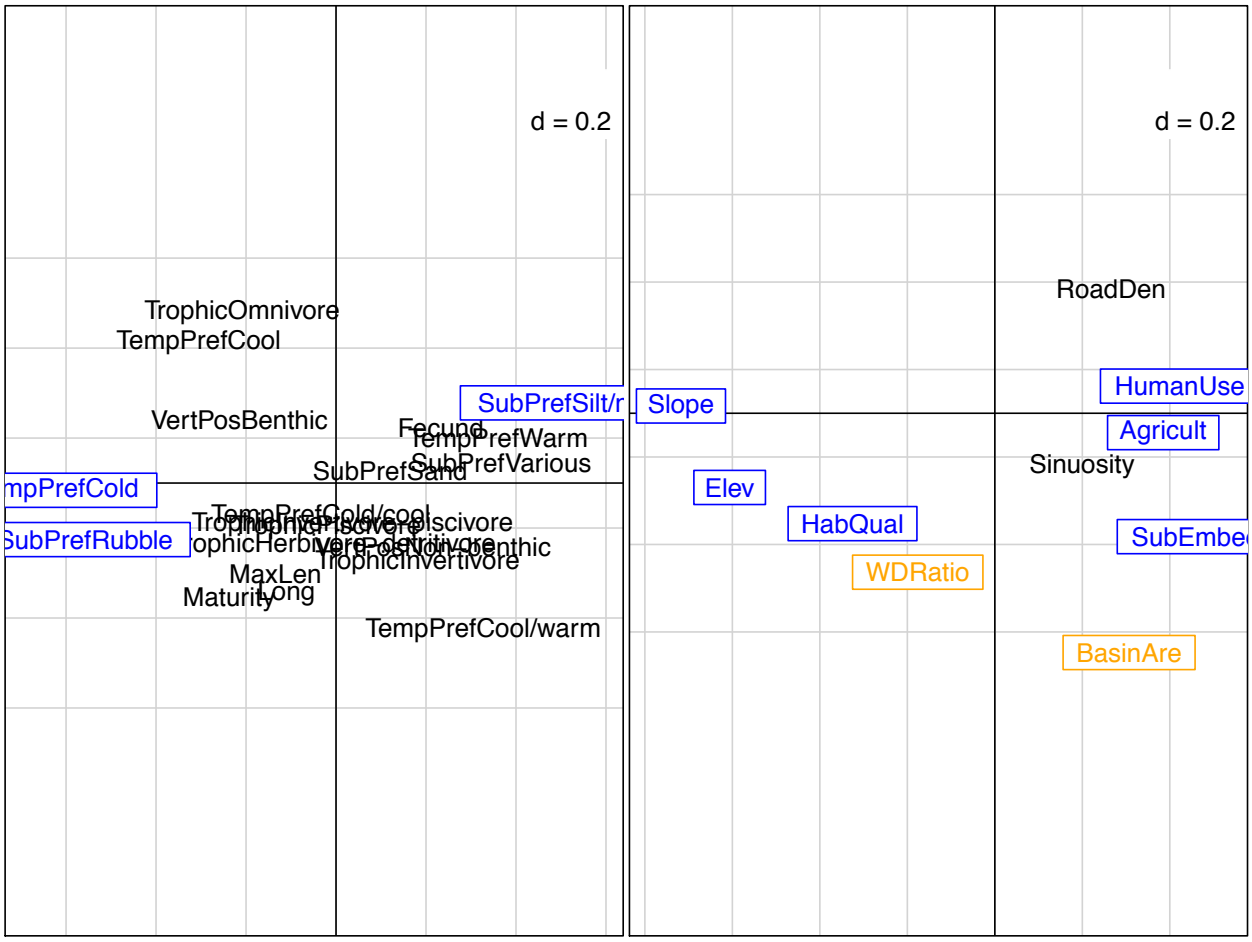
Rather than exploring the detailed tables with the results, let's just plot the p-values based on no multiple comparison correct (but you could use `p.adjust.method.D = "fdr"` if you wanted).

```
par(mfrow = c(1, 2))
plot(testQaxes, alpha = 0.05, type = "table", stat = "D2")
plot(testRaxes, alpha = 0.05, type = "table", stat = "D2")
```



You can examine at the RLQ plots presented previously to see that this test is identifying the traits and environmental variables with the largest canonical loadings in the factoral map. In fact, significance with axes can also be reported on the factorial map of RLQ analysis. Here, significant associations with the first axis are represented in blue, with the second axis in orange, with both axes in green (variables with no significant association are in black). Type:

```
par(mfrow = c(1, 2))

plot(testQaxes, alpha = 0.05, type = "biplot", stat = "D2", col = c("black",
"blue", "orange", "green"))

plot(testRaxes, alpha = 0.05, type = "biplot", stat = "D2", col = c("black",
"blue", "orange", "green"))
```



OPTIONAL READINGS

Doledec, S., Chessel, D., ter Braak, C.J.F. and Champely, S. 1996. Matching species traits to environmental variables: a new three-table ordination method. *Environmental and Ecological Statistics* **3**: 143–166.

Dray, S., Pettorelli, N., Chessel, D. 2002. Matching data sets from two different spatial samplings. *Journal of Vegetation Science* **13**: 867–874.

Dray, S. and Legendre, P. 2008. Testing the species traits-environment relationships: the fourth-corner problem revisited. *Ecology* **89**: 3400–3412.

Dray, S., P. Choler, S. Dolédec, P. R. Peres-Neto, W. Thuiller, S. Pavoine and C. J. F. ter Braak. 2014. Combining the fourth-corner and the RLQ methods for assessing trait responses to environmental variation. *Ecology* **95**: 14–21.

Legendre, P., Galzin, R. and M. L. Harmelin-Vivien. 1997. Relating behavior to habitat: solutions to the fourth-corner problem. *Ecology* **78**: 547–562.

McCune, B. 2015. The front door to the fourth corner: variations on the sample unit × trait matrix in community ecology. *Community Ecology* **16**: 267-271.

ter Braak, C., Cormont, A., Dray, S. 2012. Improved testing of species traits-environment relationships in the fourth corner problem. *Ecology* **93**: 1525–1526.