Ecological Analysis of Communities

Jeff Powell

WESTERN SYDNEY
UNIVERSITY

Hawkesbury Institute
for the Environment

July 6, 2018

Contents

Chapter 1

Ecological Analyses of Communities

1.1 Analyses of alpha diversity

Univariate linear models are used to analyse variation in alpha diversity, but there are several ways to estimate alpha diversity. The following are options for calcuating various alpha diversity metrics from a species (columns)-sample (rows) matrix or dataframe using functions in the `vegan` package.

```
# load library
library(vegan)

# species richness
specnumber(data)

# Shannon diversity
diversity(data)

# Simpson diversity
diversity(data, index='simpson')

# Pielou's evenness (Shannon diversity / log richness)
diversity(data) / log(specnumber(data))
```

See `?diversity` for more information. Note that you can calculate indices on a transposed matrix, in which species are in rows and samples in columns, using the `MARGIN = 2` argument.

Making comparisons among communities using these metrics is only sensible if the sampling effort was high enough to attain a high level of coverage (i.e., all species present within a community were likely to be observed during sampling) or communities were assessed with similar levels of sampling effort. If this is not the case, we may not have confidence that our richness estimates are meaningful.

Figure 1.1 shows an example based on high-throughput sequencing of fungal DNA isolated from sevently soil samples obtained from a *Eucalyptus saligna* plantation.

```
# load data
ixf <- read.csv('IxFsub.csv')

# load 'vegan' library
library(vegan)

## Loading required package:  permute
```

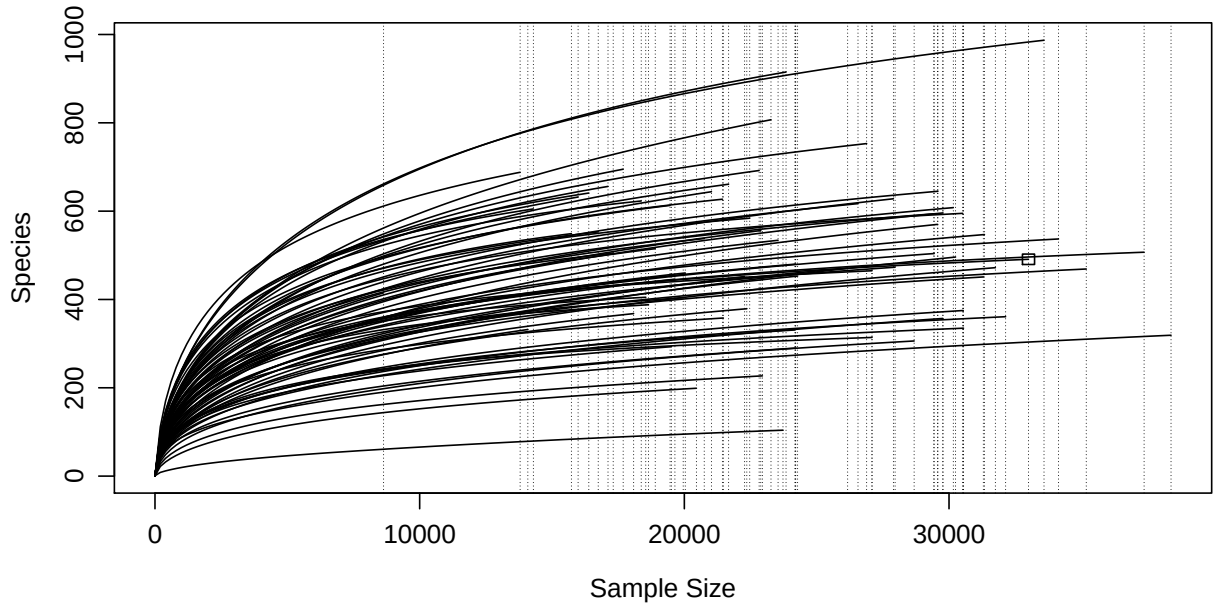

Figure 1.1: Species accumulation curves for fungal communities characterised from each of seventy soil samples.

```
## Loading required package:  lattice
```

```
## This is vegan 2.5-2
```

```
# select columns containing species data
ixf.otu <- ixf[, grep('ITSall_OTU', names(ixf))]

# show variable sampling depth
summary(rowSums(ixf.otu))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8636   19544   23634   24269   29577   38388
```

```
# plot species accumulation curves for each sample
rarecurve(ixf.otu, step=200)

# add lines indicating sampling depth
abline(v=rowSums(ixf.otu), lty='dotted', lwd=0.5)
```

We can see in Figure 1.1 that, although none of the curves are saturated, all are flattening out. Our estimates of coverage support this observation.

```
# load 'entropart' library
library(entropart)

# 'Coverage' only works on a single vector (community) at a time
# so we need to use 'apply' with MARGIN=1 to apply the function to each row

# calculate Good's coverage (only uses singletons)
summary(apply(ixf.otu, MARGIN=1, function(x){Coverage(x, Estimator='Turing')}))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9825  0.9928  0.9943  0.9941  0.9963  0.9979
```

```
# calculate Chao' coverage (uses doubletons)
summary(apply(ixf.otu, MARGIN=1, function(x){Coverage(x, Estimator='Chao')}))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9825  0.9928  0.9943  0.9941  0.9963  0.9979
```

We can extrapolate the *expected* richness in each community based on the shape of species accumulation curves and perform analyses on these extrapolated estimates. We can use the `estimateR` function to calculate extrapolated species richness. Based on Figure 1.2, in this case there is a tight relationship between observed and extrapolated richness thus both are likely to give similar results when estimating treatment effects, although the former would underestimate diversity.

```
# calculate extrapolated species richness
ixf.extrap <- estimateR(ixf.otu)

# returns a matrix with samples in columns and estimates in rows
# use 't()' to transpose matrix
# Chao and ACE estimates include mean and standard error
summary(t(ixf.extrap))
```

```
##      S.obs           S.chao1          se.chao1         S.ACE
##  Min.   :104.0   Min.   : 179.3   Min.   :13.05   Min.   : 233.2
##  1st Qu.:392.0   1st Qu.: 496.0   1st Qu.:23.57   1st Qu.: 506.2
##  Median :504.5   Median : 628.2   Median :29.61   Median : 627.9
##  Mean   :506.6   Mean   : 636.9   Mean   :29.67   Mean   : 634.2
##  3rd Qu.:611.8   3rd Qu.: 747.0   3rd Qu.:34.83   3rd Qu.: 748.8
##  Max.   :987.0   Max.   :1194.0   Max.   :58.14   Max.   :1162.7
##      se.ACE
##  Min.   : 8.721
##  1st Qu.:11.547
##  Median :12.458
##  Mean   :12.638
##  3rd Qu.:13.835
##  Max.   :17.277
```

```
# plot relationship between observed and extrapolated richness
plot(ixf.extrap['S.obs', ], ixf.extrap['S.chao1', ],
     xlab='Observed richness', ylab='Extrapolated richness')
abline(1, 1, lty='dotted')
```

If the error estimates assocated with extrapolated richness are too high, we might calculate a rarefied richness from our community matrix based on a constant effort across all samples using the `rarefy` function.

```
# use the 'rarefy' function to calculate richness at the lowest sampling depth
ixf.rare <- rarefy(ixf.otu, min(rowSums(ixf.otu)))
summary(ixf.rare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.9   299.5   360.0   377.3   469.0   661.7
```

```
# plot relationship between observed and rarefied richness
plot(specnumber(ixf.otu), ixf.rare,
     xlab='Observed richness', ylab='Rarefied richness')
abline(1, 1, lty='dotted')
```

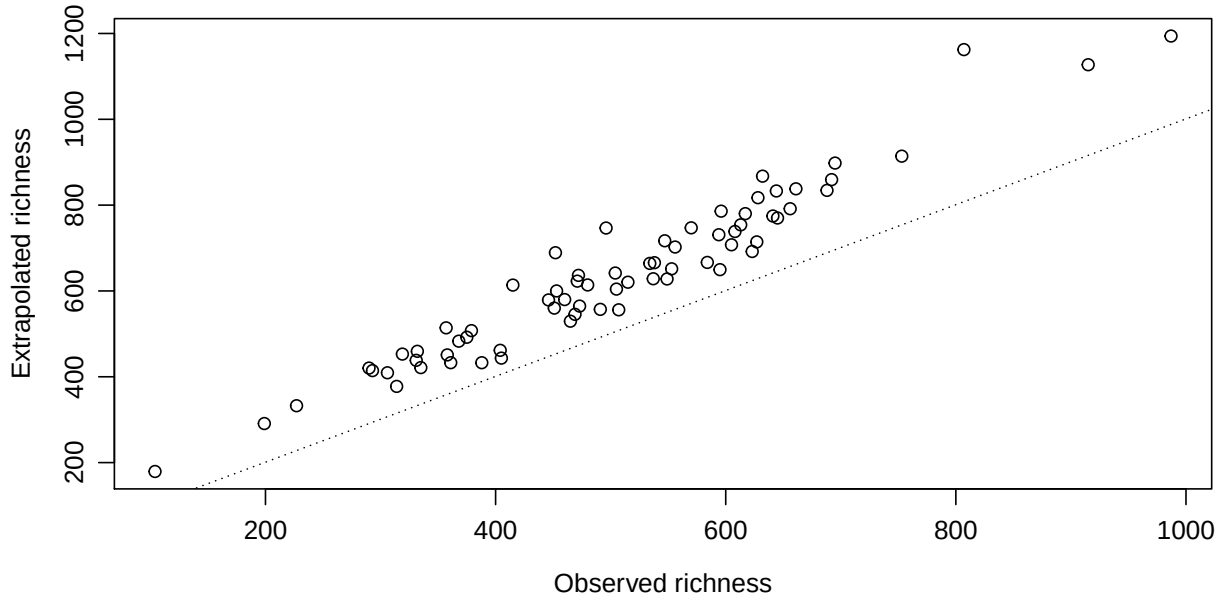

Figure 1.2: Comparison between observed and extrapolated species richness in fungal communities from seventy soil samples. The dotted line indicates the 1:1 relationship.

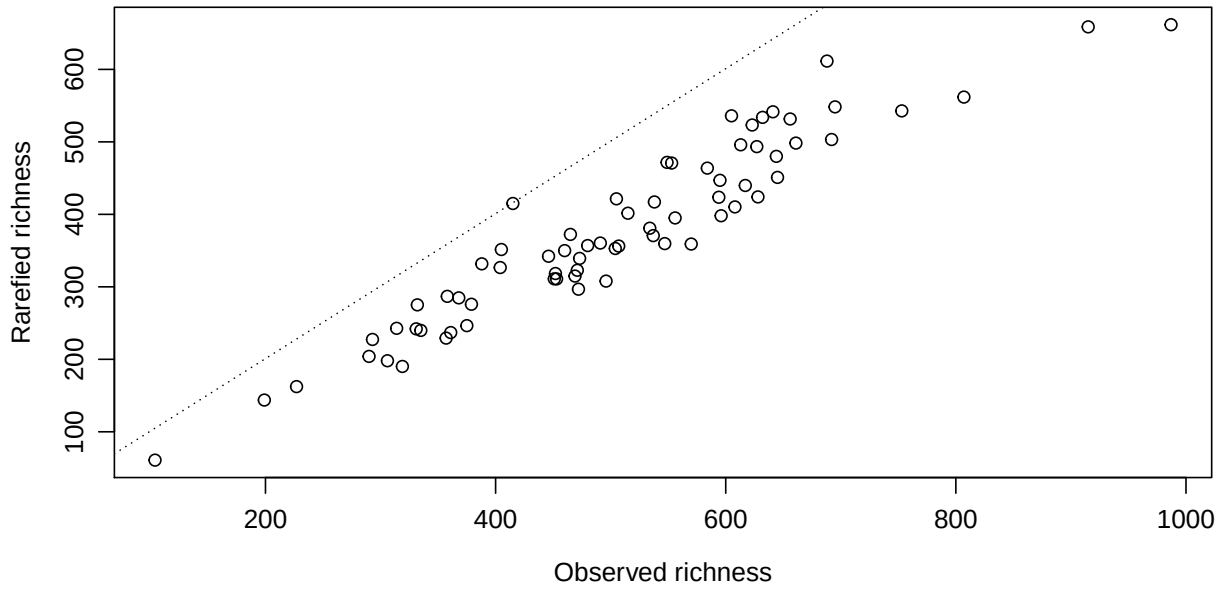

Figure 1.3: Comparison between observed and rarefied species richness in fungal communities from seventy soil samples. The dotted line indicates the 1:1 relationship.

1.2 Analyses of beta diversity

1.2.1 What is the best multivariate analysis method to use?

When analysing multivariate data, you should think very carefully about what it is that you actually want to do. For instance, do you want to know whether there are patterns in the data, what are the important drivers generating patterns in the data, or which variables best represent patterns in the data? Once you have answered this question, it will become much easier to narrow down the possible approaches and identify the specific approach that is best suited to your data.

The choice of analysis can be determined by answering a series of questions:

- Am I looking for patterns in the data or do I want to make predictions about particular species and the ways in which they covary? For the former, use ordination methods (Section 1.3). For the latter, use multivariate ANOVA or GLM aproaches (Section 1.4.5.3).
- For ordination, are my response variables continuous and approximately linear? If so, approaches based on principal components analysis (PCA; Section 1.3.1) are appropriate. If not, PCA may still work but this depends on gradient length, meaning the degree of overlap in the response variables among samples (e.g., if species composition, or betadiversity, does not change very much among samples; Section 1.3.2.1).
- For nonlinear and/or discrete responses, what might be an appropriate way to calculate the degree that the responses among samples differ? Raw count data may be used to calculate Chi-squared distances and perform correspondence analysis (CA; Section 1.3.2). Other data types (e.g., proportions and relative abundances) can be used to calculate various distance indices for principal coordinates analysis (PCoA; Section 1.3.3) or converted to ranks for nonmetric multidimensional scaling (NMDS; Section 1.3.4).
- Do I have additional data that may provide insight into the degree of resemblance among samples? If so, use constrained ordination (e.g., redundancy analysis [RDA], canonical correspondence analysis [CCA]) to partition variation in the response variables to these additional predictor variables prior to determining how much variation remains unexplained (Section 1.4.2).
- Do I want to estimate and compare the importance of specific predictor variables and their interactions? If so, use permutation-based approaches such as permutational MANOVA (PerMANOVA) to test specific hypotheses (Section 1.4.5.1).

Further reading Mike Palmer maintains a website (http://ordination.okstate.edu) that is helpful to get an overview of the many approaches that are available and the definitions of terms related to these approaches. The *"GUSTA ME"* project (GUide to STatistical Analysis in Microbial Ecology; http://mb3is.megx.net/gustame/home) includes accessible descriptions of multivariate statistical methods and worked examples (including flow charts) from the microbial ecology literature. *"The R Book"* by Michael Crawley includes a chapter on these methods that is also accessible to environmental scientists. For detailed discussion, see *"Numerical Ecology"* by Legendre and Legendre. There are several resources for performing multivariate statistical analysis in **R** and a summary of these can be found at the Environmetrics CRAN Task View (http://cran.r-project.org/web/views/Environmetrics.html) under the "Ordination", "Dissimilarity coefficients", and "Cluster analysis" headings and at the Multivariate CRAN Task View (https://cran.r-project.org/web/views/Multivariate.html). A few tools are in the 'stats' package, which comes with your **R** distribution, but environmental scientists tend to mainly use the `vegan`, `ade4`, or `labdsv` packages. There is a lot of overlap between these packages in terms of the types of analyses that can be done, but the structure of the output is very different. Both `vegan` and `ade4` contain a function named `cca()`, so be careful if you have both packages loaded at the same time since the most recently loaded version of `cca()` will be called. You can specify the package in the call to the function – for example, `vegan::cca()`. Many other packages used by ecologists and evolutionary biologists rely on these packages. The Spatial CRAN Task View (http://cran.r-project.org/web/views/Spatial.html) will also be useful for more complex analysis of univariate and multivariate data with explicit spatial structure.

1.2.2 Do you need to modify your community matrix?

If coverage varies among samples and particularly if coverage is low, we might be concerned that we are overestimating betadiversity and possibly increasing our chances of drawing incorrect conclusions. In cases like this, some recommend randomly resampling the community data at a common level of sampling effort. For the fungal community data generated from the *Eucalyptus saligna* plantation, we might argue that resampling isn't necessary because all of our samples have fairly good coverage and, therefore, we have captured most of the species present in each replicate. However, for the sake of convincing the second Reviewer, let's prepare a resampled community dataframe for subsequent analyses.

```
# randomly resample the community matrix at the lowest effort
ixf.sub <- rrarefy(ixf.otu, min(rowSums(ixf.otu)))

# confirm that sampling effort is the same in the resulting object
summary(rowSums(ixf.sub))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8636    8636    8636    8636    8636    8636
```

Note that each time this is run, a new random result will likely be observed. Therefore, this object should be saved if you wish to repeat any analyses.

You may also wish to modify your community data to eliminate potentially spurious observations or retain only those observations that clear a pre-defined threshold. This is easy to do once the community table is converted to a matrix, allowing for comparison of inferences based on community data that have or have not been cleaned up.

```
# create matrix from community dataframe
ixf.mat <- as.matrix(ixf.otu)

# original number of observations in total
```

```r
sum(ixf.mat)
```

```
## [1] 1698855
```

```r
# original number of observations per sample
summary(rowSums(ixf.mat))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8636   19544   23634   24269   29577   38388
```

```r
# original number of observations per species
summary(colSums(ixf.mat))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      1.0      3.0      9.0    307.4     44.0 110287.0
```

```r
# remove singletons
ixf.sing <- ixf.mat
ixf.sing[ixf.mat == 1] <- 0
summary(rowSums(ixf.sing))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8485   19443   23493   24137   29417   38284
```

```r
summary(colSums(ixf.sing))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      0.0      2.0      8.0    305.7     41.5 110286.0
```

```r
# remove observations that are less than 0.1% of a sample
ixf.swap <- ixf.mat
ixf.swap[ixf.mat / rowSums(ixf.mat) < 0.001] <- 0
summary(rowSums(ixf.swap))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7890   18263   21922   22268   27170   37140
```

```r
# remove species with fewer than 10 observations
ixf.ltten <- ixf.mat[, colSums(ixf.mat) >= 10]
summary(colSums(ixf.ltten))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     10.0     19.0     47.0    629.1    178.0 110287.0
```

```r
# remove species in fewer than five samples
ixf.ltfive <- ixf.mat[, colSums(decostand(ixf.mat, method = 'pa')) >= 5]
summary(colSums(decostand(ixf.ltfive, method = 'pa')))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00    7.00   10.00   15.81   19.00   70.00
```

```r
# remove species that make up less than 0.005% of all observations
ixf.rare <- ixf.mat[, colSums(ixf.mat) / sum(ixf.mat) >= 0.00005]
summary(colSums(ixf.rare))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     85.0    147.0    305.5   1647.4    916.8 110287.0
```

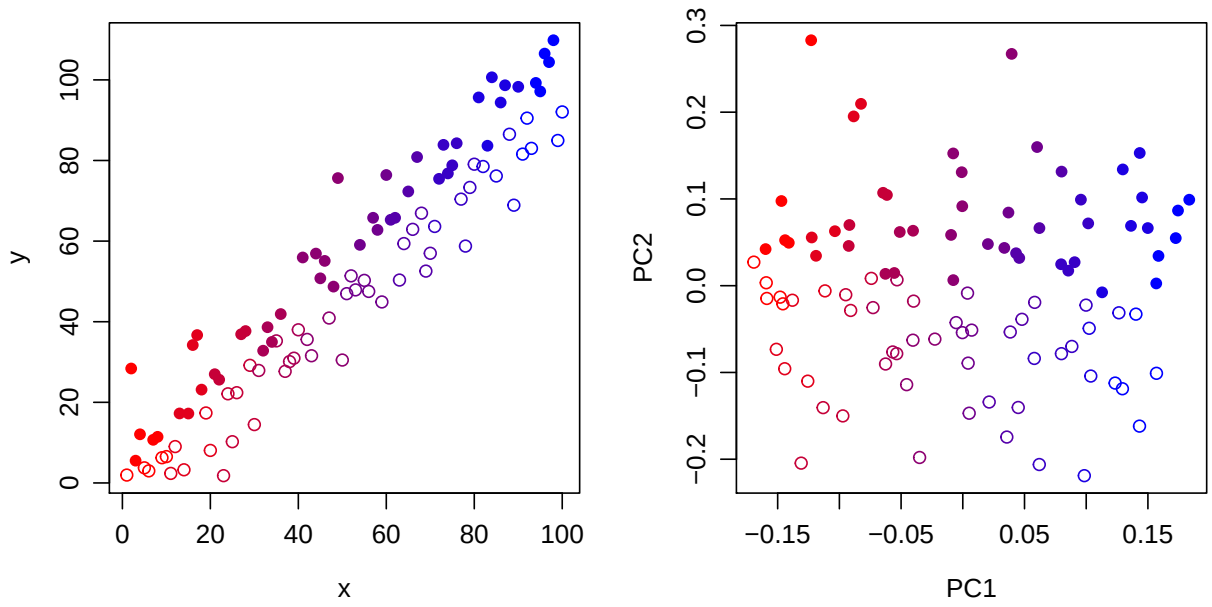

Figure 1.4: Example of two correlated variables (raw data shown on the left) and the resulting transformation following PCA (panel on right). The individual points are filled and coloured identically in both panels so that one can see the relationship between their position in each panel.

1.3 Ordination: Exploring your multivariate data

Unconstrained ordination approaches are very useful for simplifying multivariate data to visualise patterns. These should always be the first step in the analysis of multivariate data, even when you are interested in testing specific hypotheses regarding the potential indicators and drivers of data structure.

1.3.1 Principal Components Analysis (PCA)

PCA is ultimately the transformation of continuous multivariate data into new variables that maximise the amount of variance explained in the data, with each subsequent variable orthogonal (think perpendicular but in greater than two dimensions) to the previous variable and explaining decreasing amounts of variation. The new variables are linear combinations of the original variables.

Figure 1.4 shows an example of PCA, in which two correlated variables are transformed so that the first axis corresponds to the linear relationship between the two (represented by the shifting colours) and the second corresponds to the remaining variation (represented by the filled/unfilled circles). The individual points and the axes from the left panel are rotated in position to their locations on the right panel; their loadings reflect the degree of rotation.

Performing these is simple using `prcomp`, but we will use `rda` in `vegan` for the sake of consistency (particularly valuable when showing plotting methods). We'll use the 'varechem' data from the `vegan` package, which contains observations of soil variables associated with 24 sites grazed by reindeer.

```
## library(vegan)  # loads the 'vegan' library

data(varechem)  # read data included in the package into your workspace
str(varechem)  # observe the object structure
```

```
## 'data.frame': 24 obs. of  14 variables:
##  $ N       : num  19.8 13.4 20.2 20.6 23.8 22.8 26.6 24.2 29.8 28.1 ...
##  $ P       : num  42.1 39.1 67.7 60.8 54.5 40.9 36.7 31 73.5 40.5 ...
##  $ K       : num  140 167 207 234 181 ...
##  $ Ca      : num  519 357 973 834 777 ...
##  $ Mg      : num  90 70.7 209.1 127.2 125.8 ...
##  $ S       : num  32.3 35.2 58.1 40.7 39.5 40.8 33.8 27.1 42.5 60.2 ...
##  $ Al      : num  39 88.1 138 15.4 24.2 ...
##  $ Fe      : num  40.9 39 35.4 4.4 3 ...
##  $ Mn      : num  58.1 52.4 32.1 132 50.1 ...
##  $ Zn      : num  4.5 5.4 16.8 10.7 6.6 9.1 7.4 5.2 9.3 9.1 ...
##  $ Mo      : num  0.3 0.3 0.8 0.2 0.3 0.4 0.3 0.3 0.3 0.5 ...
##  $ Baresoil: num  43.9 23.6 21.2 18.7 46 40.5 23 29.8 17.6 29.9 ...
##  $ Humdepth: num  2.2 2.2 2 2.9 3 3.8 2.8 2 3 2.2 ...
##  $ pH      : num  2.7 2.8 3 2.8 2.7 2.7 2.8 2.8 2.8 2.8 ...
```

```
# the variables differ in the scales of their variances
var(varechem$S)  # variance in sulphur concentration
```

```
## [1] 136.1382
```

```
var(varechem$Ca)  # variance in calcium concentration
```

```
## [1] 59332.17
```

The values associated with the different gradients are very different; for example, sulphur varies on a much smaller scale than calcium and their variances have to be standardised otherwise the large absolute variance associated with calcium will have much greater weight than that for sulphur. This is done with the `scale` argument (note that `scale=FALSE` is the default).

```
chem.pca <- rda(varechem, scale=TRUE)  # use 'scale=TRUE' to standardise variances
plot(chem.pca)  # plot the resulting object
```

The plot (Fig. 1.5) shows how the samples (indicated by number) are separated based on the first two principal components, the new variables resulting from the transformations of the original variables based on their ability to account for variation in the multivariate data. The samples are labelled according to the row names of the input dataframe or matrix. The black numbers indicate the loading of the individual samples ('sites') and the red labels indicate the loadings associated with the original variables ('species'; it is convention to refer to the columns in the multivariate response table as 'species', which becomes important when extracting these loadings). From this, we can see that the calcium and magnesium overlap entirely and, therefore, are positively correlated along both of the first two principal components. Because they are almost horizontal, their loadings associated with the first axis are much greater than that for the second axis. Sulfur and nitrogen are negatively correlated along both axes. Sample numbers positioned adjacent to variables along each axis indicates samples in which the values for those variable are high; from this we see that sample 24 is relatively high in calcium, magnesium, zinc, potassium, phosphorus, and sulfur while samples 3, 5, 6, and 7 are low in these elements.

```
varechem[c('5','6','7','3','24'),c('Ca','Mg','Zn','K','P','S')]

summary(chem.pca, display=NULL)  # use 'display=NULL' to suppress loadings tables
# (output not shown - run this yourself)
```

According to the summary table, these components explain 37 and 23 % of the variance in these data. The third component explains 12 % of variation, and we may be interested to see how samples and variables are distributed along this axis. To do this, we use the `choice` argument.

The first principal component is still plotted on the x-axis of Fig. 1.6, but now the third component

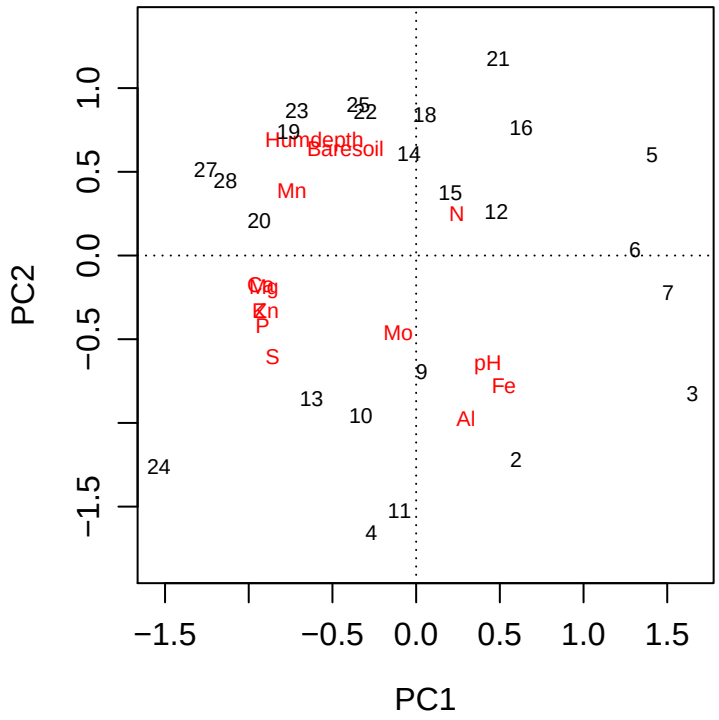

Figure 1.5: Biplot showing ordination of sites based on PCA of soil variables.

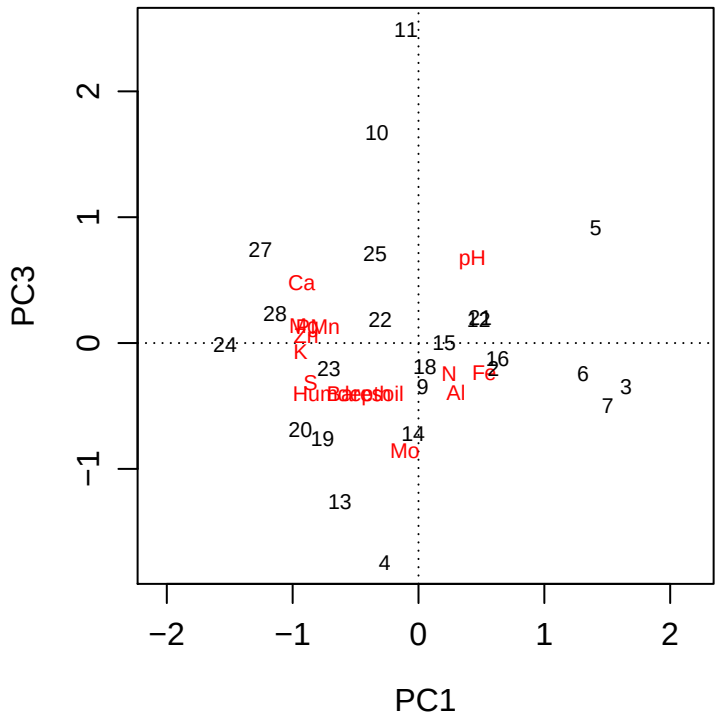

Figure 1.6: Biplot showing ordination (first and third principal components) of sites based on PCA of soil variables.

is plotted on the y-axis. We can see that calcium and magnesium are separated along the third axis (variation in calcium is explained by this axis, but not magnesium, which has loading close to zero). We also see that nitrogen and sulfur are positively correlated along the third axis (both are positioned on the negative side of this axis). The sample and variable scores for the first component are the same as before.

Most of the variation is accounted for in the first two principal components and $> 90\%$ is accounted for by the first six components. Any patterns observed associated with subsequent components will not contain much information.

Try this yourself Try to understand the importance of scaling variances by setting `scale = F` and looking at the result.

1.3.2 Correspondence analysis (CA)

PCA uses euclidean distances, geometrical distances in multidimensional space, to estimate the divergence between samples. This is appropriate for continuous variables exhibiting normal distributions, like the above example, but not for data that resemble frequencies or counts. CA uses a different approach to estimate divergence among samples, based on calculation of χ^2 distances as for contingency tables. This property makes CA a better option than PCA for analysing tables of species counts in different environments.

CA is performed using `cca()` in `vegan` by providing only the response matrix as an argument to the function. We will use the plant community data collected from the same locations as the soil data analysed above; these data are stored in `varespec` and consist of observations relating to the percent cover associated with 44 plant species.

```
data(varespec)  # read data included in the package into your workspace
spec.ca <- cca(varespec)  # performs CA on single input matrix
plot(spec.ca)  # plot resulting object
```

```
summary(spec.ca, display=NULL)  # use 'display=NULL' to suppress loadings tables
```

```
##
## Call:
## cca(X = varespec)
##
## Partitioning of scaled Chi-square:
##               Inertia Proportion
## Total           2.083          1
## Unconstrained   2.083          1
##
## Eigenvalues, and their contribution to the scaled Chi-square
##
## Importance of components:
##                            CA1    CA2    CA3     CA4     CA5     CA6     CA7
## Eigenvalue            0.5249 0.3568 0.2344 0.19546 0.17762 0.12156 0.11549
## Proportion Explained  0.2520 0.1713 0.1125 0.09383 0.08526 0.05835 0.05544
## Cumulative Proportion 0.2520 0.4233 0.5358 0.62962 0.71489 0.77324 0.82868
##                            CA8     CA9    CA10    CA11    CA12    CA13
## Eigenvalue            0.08894 0.07318 0.05752 0.04434 0.02546 0.01710
## Proportion Explained  0.04269 0.03513 0.02761 0.02129 0.01222 0.00821
## Cumulative Proportion 0.87137 0.90650 0.93411 0.95539 0.96762 0.97583
##                           CA14     CA15     CA16     CA17     CA18
```

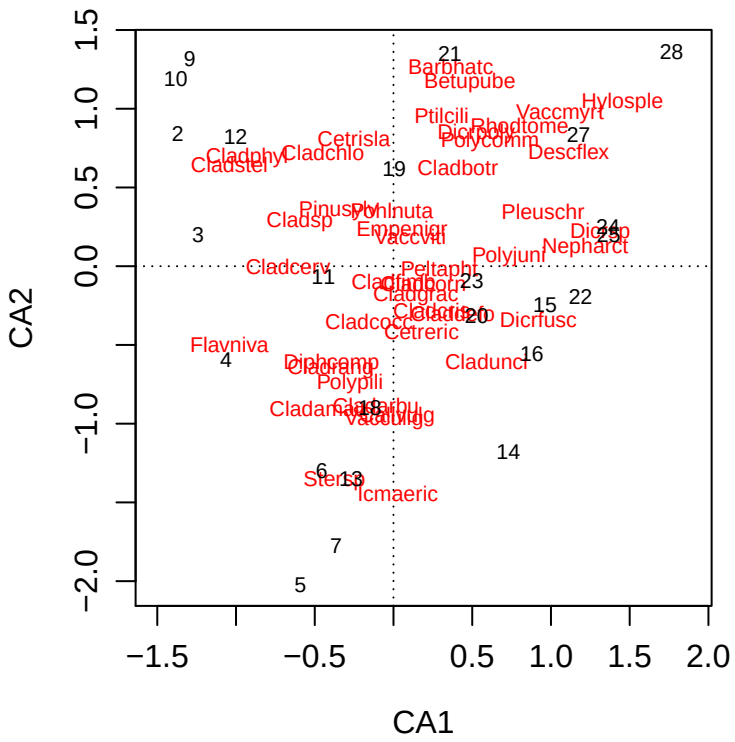

Figure 1.7: Biplot showing ordination of sites based on plant communities following CA.

```
## Eigenvalue            0.014896 0.010160 0.007830 0.006032 0.004008
## Proportion Explained  0.007151 0.004877 0.003759 0.002896 0.001924
## Cumulative Proportion 0.982978 0.987855 0.991614 0.994510 0.996434
##                          CA19      CA20      CA21      CA22      CA23
## Eigenvalue            0.002865 0.0019275 0.0018074 0.0005864 0.0002434
## Proportion Explained  0.001375 0.0009253 0.0008676 0.0002815 0.0001168
## Cumulative Proportion 0.997809 0.9987341 0.9996017 0.9998832 1.0000000
##
## Scaling 2 for species and site scores
## * Species are scaled proportional to eigenvalues
## * Sites are unscaled: weighted dispersion equal on all dimensions
```

The plot in Fig. 1.7 is interpreted in the same way as the PCA biplot: the numbers represent the scores of the samples on the first two correspondence axes and the labels in red represent the loadings associated with the species. The summary output is interpreted in the same way as the output from PCA.

There was no scaling of the variables in our example but this may be desirable if, for example, the species vary greatly in total abundance along the whole gradient. The `decostand` function will transform the input matrix into a standarised form with the desired characteristics. There are several methods available in decostand but the most commonly used are `standardize` (analogous to `scale=T`, but results in an error when using `cca` due to negative values), `normalize` (results in row sums of squares equal to one), `total` (returns proportional relative abundances within each sample), and `hellinger` (as for `total` but then returns the square root of the result).

In Fig. 1.8, the plot on the left shows the result of CA on the untransformed species abundances. The figure on the right shows the result when the data have been normalized (using `cca(decostand(varespec, method='normalize'))`). The spread of the site scores is slightly less triangular on the right than on the

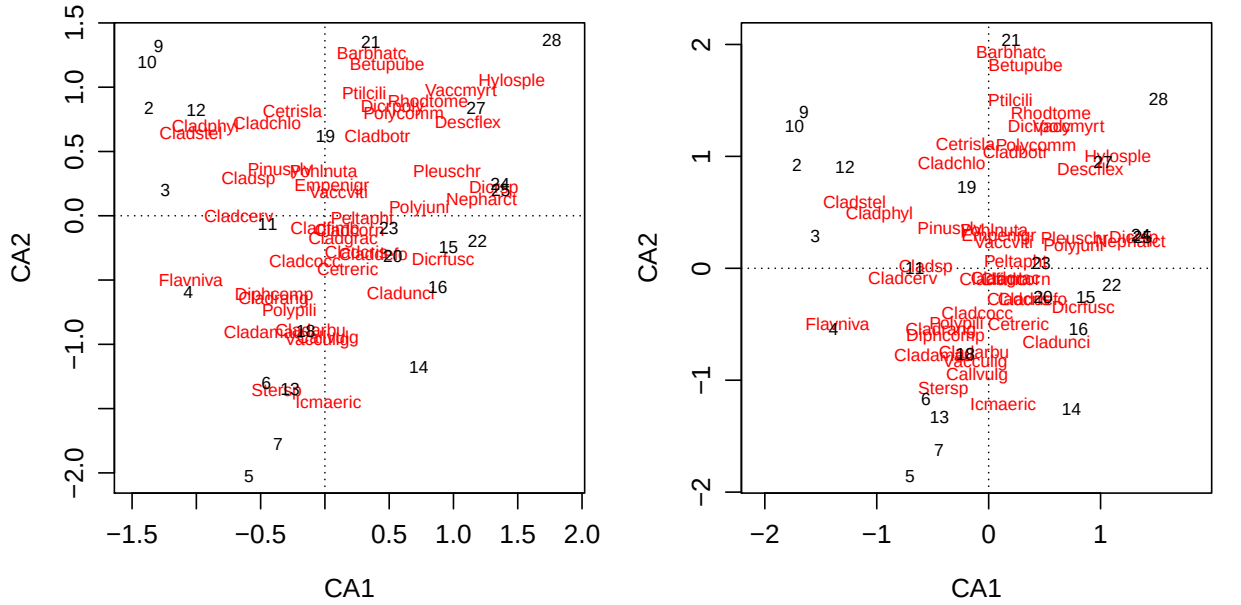

Figure 1.8: Correspondence analysis of community matrix based on raw data (left) and normalized data (right).

left, suggesting that the transformed data are not as distorted and the analysis is more likely to result in meaningful patterns.

1.3.2.1 PCA or CA?

For analysis of multivariate species data, PCA (and RDA, see Section 1.4.2) is generally used when there is low turnover is species composition among samples while CA (and CCA, see Section 1.4.2) is generally used when turnover is high. One can determine which is the case by estimating the length of the gradient in species composition using the `decorana()` function.

```
spec.dca <- decorana(varespec)
summary(spec.dca, display='none')  # note different argument for 'display'

##
## Call:
## decorana(veg = varespec)
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                   DCA1   DCA2    DCA3    DCA4
## Eigenvalues     0.5235 0.3253 0.20010 0.19176
## Decorana values 0.5249 0.1572 0.09669 0.06075
## Axis lengths    2.8161 2.2054 1.54650 1.64864
```

The gradient length along the first axis is 2.8161. A rule of thumb is that, since this value is less than 3, PCA would be an appropriate method (use CCA when this value is greater than 3). However, it is not much smaller than 3; we could also try transforming the data using an alternative distance index prior to analysing them (next section).

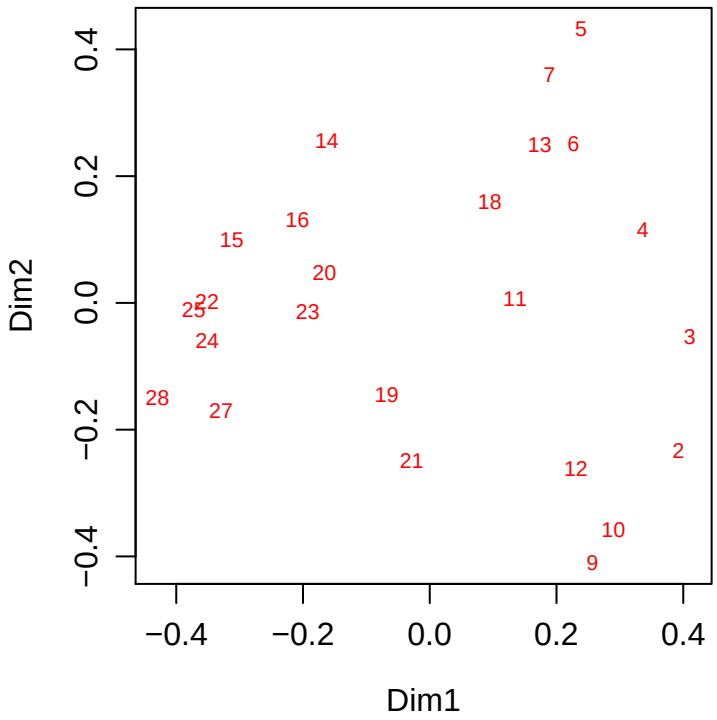

Figure 1.9: Principal coordinates analysis of the vegetation community matrix, using Bray-Curtis dissimilarities.

1.3.3 Principal coordinates analysis (PCoA)

Just as euclidean distances are not appropriate for estimating the divergence between samples when analysing frequency data, some data types require an alternative to χ^2 distances. PCoA allows for the analysis of these data using distance matrices estimated from various multidimensional scaling indices. Many common metrics for estimating community dissimilarity are available in the `vegdist()` function (see the help page for these indices and their formulae). The distance matrix is used as the input for `wcmdscale()` in `vegan`, which performs the PCoA.

```
# use the default index ('bray') and 'eig=TRUE' to save the calculated eigenvalues
spec.pco <- wcmdscale(vegdist(varespec), eig=TRUE)
plot(spec.pco)
```

Notice in Fig. 1.9 that the ordinated data aren't as distorted as they were when using PCA , suggesting that we could interpret the patterns in these data in a meaningful way (although they do look a bit like the outline of Australia!). Also notice that there are no labels indicating the species loadings; this is because this information is not retained in the distance matrix that was provided as input. There is no built-in `summary()` function for returning interpretable output from PCoA analysis, but we can look at the eigenvalues to see how much variation is accounted for by the different principal coordinate axes (these values are available since we specified `eig=TRUE`).

```
# Return the eigenvalues
spec.pco$eig
```

```
##  [1]  1.7552165397  1.1334455380  0.4429018480  0.3698054310  0.2453531540
##  [6]  0.1960920773  0.1751130911  0.1284466728  0.0971594360  0.0759600747
## [11]  0.0637177905  0.0583225124  0.0394933793  0.0172699235  0.0051011077
## [16] -0.0004131222 -0.0064653552 -0.0133147491 -0.0253943546 -0.0375104890
```

```
## [21] -0.0480068852 -0.0537145779 -0.0741390257
```

```
# Return the proportion of variance explained (non-negative eigenvalues)
eigens <- spec.pco$eig[spec.pco$eig >= 0]
# proportion explained by each axis
eigens / sum(eigens)
```

```
##  [1] 0.365411388 0.235967413 0.092205933 0.076988288 0.051079075
##  [6] 0.040823612 0.036456082 0.026740790 0.020227228 0.015813819
## [11] 0.013265147 0.012141926 0.008221966 0.003595355 0.001061979
```

```
# cumulative proportion explained
cumsum(eigens / sum(eigens))
```

```
##  [1] 0.3654114 0.6013788 0.6935847 0.7705730 0.8216521 0.8624757 0.8989318
##  [8] 0.9256726 0.9458998 0.9617136 0.9749788 0.9871207 0.9953427 0.9989380
## [15] 1.0000000
```

The first two values (1.75, 1.13) are much larger than the rest, suggesting that most of the variation is accounted for by these two axes. There are also negative eigenvalues; to estimate the relative proportions of variance explained, we only use the nonnegative eigenvalues (indexing can be used to select these, as shown above).

How does one choose one of the available indices for calculating dissimilarity? One way to do so is to pick the index that provides the best rank-order similarity (i.e., nonparametric correlation based on two rank-transformed variables) to the gradient under study, using `rankindex()`. This gradient might be represented in an environmental matrix (here we will use the soil chemistry data from above) or some other indicator of a putative gradient (e.g., sampling coordinates).

```
# For rankindex(), the first argument is the gradient,
# the second is the community matrix. The output is a named vector of rank-
# -order similarities, each representing a dissimilarity index.

# First calculate rank-order similarity on the raw community matrix
rankindex(scale(varechem), varespec)
```

```
##       euc       man       gow       bra       kul
## 0.2396330 0.2735087 0.2288358 0.2837910 0.2839834
```

```
# Then calculate rank-order similarity after applying the
# hellinger transformation (see ?decostand)
rankindex(scale(varechem), decostand(varespec, method='hellinger'))
```

```
##       euc       man       gow       bra       kul
## 0.2842265 0.2562909 0.2570694 0.3183687 0.3168585
```

This function can also be used to determine which standardisation approach should be applied to the data. Comparing the values returned by the two calls to `rankindex` immediately above, we see that the Bray-Curtis index (`'bra'`) provides a higher rank-order similarity than the other tested indices when the `hellinger` transformation is applied to the community matrix.

Try this yourself Try using different dissimilarity indices to see the effect this has on the PCoA result.

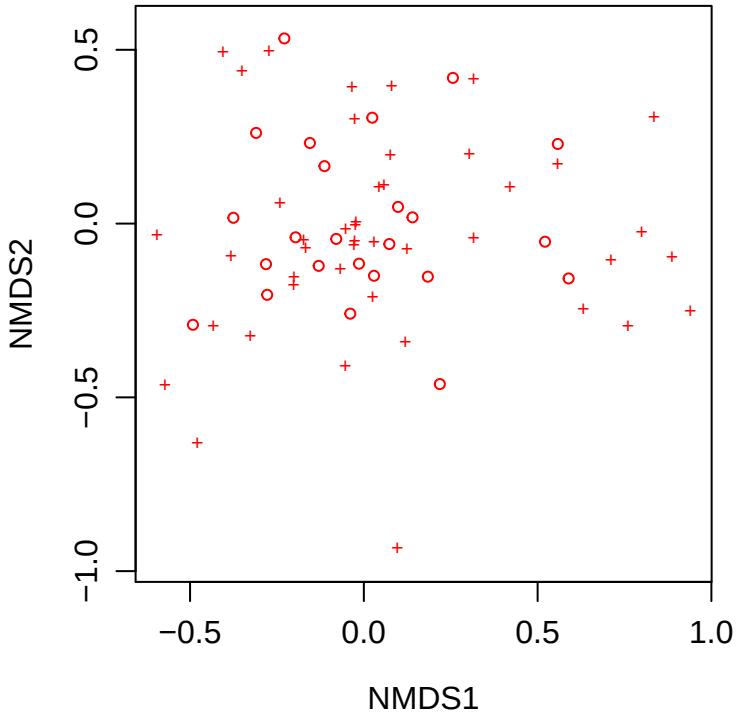

Figure 1.10: NMDS ordination of sites based on variation in plant communities

1.3.4 Non-metric multidimensional scaling (NMDS)

NMDS maximises the differences between samples on few dimensions, which can be particularly useful for visual representation of the dissimilarities between samples. It is "nonmetric" because the data undergo rank-order transformation and their positions are moved during the procedure to minimise stress. The `metaMDS` function in `vegan` performs NMDS on either a table of community data or a distance matrix calculated from this table, as above. Since the purpose of this approach is to visualise the spread of the data in reduced dimensions, it makes sense to plot the data (Fig. 1.10).

```
spec.nmds <- metaMDS(varespec,trymax=40)  # (output not shown)
plot(spec.nmds)
```

The procedure is iterative, and technical data are output to the screen as it runs. `*** Solution reached` should appear at the end of this output (note that the output above is abbreviated). If not, redo the analysis and increase the value associated with the `trymax` argument (the default is 20). The stress in the final run is 0.183, which is acceptable; stress values greater than 0.3 are unsatisfactory suggesting that the dissimilarities are not effectively captured by these two dimensions. Increase the number of dimensions using the `k` argument (the default is 2).

1.3.4.1 Manipulating graphics from *vegan* objects

Figure 1.10 shows the loadings for the sites (circles) and species (crosses) from the above NMDS analysis. This is a good point to show how graphics can be manipulated as no labels are included in the plot. While the plotting functions in `vegan` are useful to visualise output from `rda` and `cca` and for interpreting this output, they are very basic and the results are not necessarily of publication quality. However, we can use our output and **R**'s default plotting functions to modify the graphics as we did in chapter **??**.

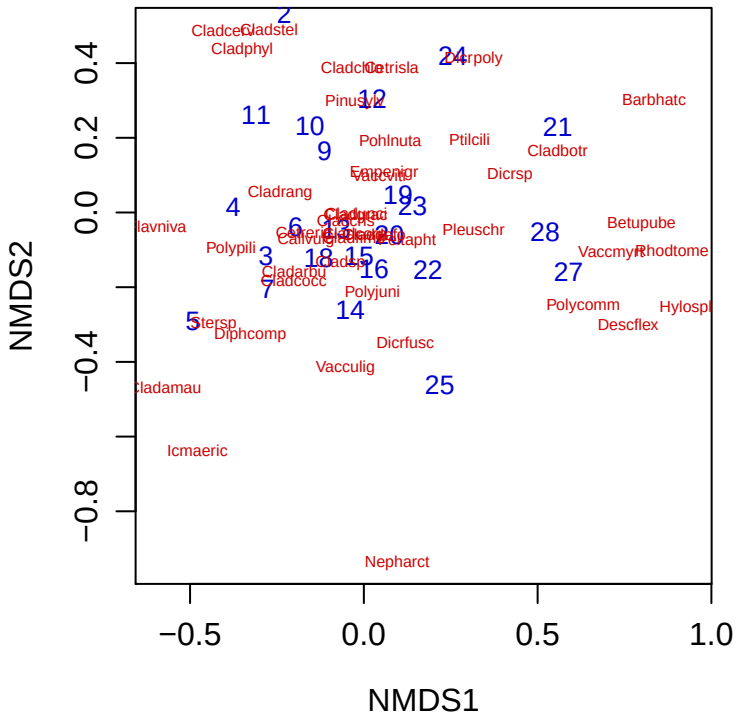

Figure 1.11: NMDS ordination of sites based on variation in plant communities.

The information needed to generate a plot for publication can be found using the `scores` function, which returns 'site' and 'species' loadings based on which is specified using the `display` argument.

```
# Set up plot window
plot(scores(spec.nmds, display='species'), type='n')

# Sites loadings
text(scores(spec.nmds, display='sites'),
     labels=rownames(scores(spec.nmds, display='sites')),
     cex=0.8, col="blue3")

# Species loadings
text(scores(spec.nmds, display='species'),
     labels=rownames(scores(spec.nmds, display='species')),
     cex=0.5, col="red3")
```

This was a multi-step process. First we set up the plot window, including the figure axes, but suppressed the plotting of the data points using `type='n'`. We used the `species` loadings to set up the plot window in this case as these expand further along both axes than the `sites` loadings; had we used the `sites` loadings, some of the labels belonging to species would fall outside the plot area. Then we used `text()` to plots the row names for the table of sites loadings in place of the data points. The second call to `text()` plots the row names for the table of species loadings; we used the `cex` argument in each line to reduce the text size.

Try this yourself Using the examples from PCA, CA, and PCoA, plot the results and manipulate. Note that for PCoA, no species loadings are provided as the analysis is performed on a distance matrix, not species abundances.

1.4 Two (or more)-table analysis: Unveiling drivers of data structure

Up to this point, we have used ordination to look for structure in multivariate data. Once that structure is observed, you may have ideas about what factors are driving that structure and would like to test the hypotheses that these factors are actually important. To do this is to apply a constraint to the data, partitioning variation in the data to specific factors, or linear combinations of these specific factors (as above), and then comparing this partitioned variation to any remaining variation in the data.

1.4.1 Matrix correlations

Given two tables, each containing multivariate data about two aspects of the system under study (for example, species abundances and environmental characteristics), it may be of value to determine whether the responses across the two tables are related. In ecology, it is common to have collected data on species abundances in a variety of environments and to try to explain variation in those species' distributions due to characteristics of their environments. Similarly, physiologists and evolutionary biologists often take multiple types of measurements on a number of organisms and may wish to determine whether aspects of these organisms or their environment are important predictors of their responses.

Correlations are estimated on dissimilarity matrices, not on the raw data, so it is necessary to transform each table of data using `vegdist()` as for PCoA above.

```
mantel(vegdist(varespec,'bray'),vegdist(scale(varechem),'euclidian'))
```

```
##
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = vegdist(varespec, "bray"), ydis = vegdist(scale(varechem),      "euclidian"))
##
## Mantel statistic r: 0.3047
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.109 0.142 0.171 0.188
## Permutation: free
## Number of permutations: 999
```

Note that we used different indices for the two tables. Bray-Curtis dissimilarities were estimated for the species-sample table, while euclidean distances were estimated for the table containing soil variables (after standardising these data using the `scale()` function. The function provides a correlation coefficient and an outcome of the hypothesis test that the coefficient does not differ from zero, by permutation. Here we see that the two matrices are correlated, which may mean that the variables in one table are important drivers of the values in the other or that both types of variables share a common driver or set of drivers.

1.4.2 Canonical analysis

By combining ordination techniques and multiple linear regression, we can use regression approaches to explain variation in one multivariate data table by variation in another multivariate data table from observations of the same objects. Prior to the availability of suitable software programs, principal components were extracted from the ordination of the explanatory matrix and then related to the ordination of the response matrix; this approach is named indirect gradient analysis. This approach has gradually been replaced by canonical analysis, or direct gradient analysis, in which the explanatory matrix is directly involved in the ordination of the response matrix.

1.4.2.1 Redundancy analysis (RDA)

RDA is an extension of PCA and, as such, is appropriate for estimating the importance of constraining variables along short gradients that display low species turnover (recall Section 1.3.2.1). Calculations are performed using the `rda()` function in `vegan`.

```
vare.rda <- rda(varespec,scale(varechem), scale=T)
vare.rda
```

```
## Call: rda(X = varespec, Y = scale(varechem), scale = T)
##
##               Inertia Proportion Rank
## Total         44.0000     1.0000
## Constrained   28.5273     0.6483   14
## Unconstrained 15.4727     0.3517    9
## Inertia is correlations
##
## Eigenvalues for constrained axes:
##  RDA1  RDA2  RDA3  RDA4  RDA5  RDA6  RDA7  RDA8  RDA9 RDA10 RDA11 RDA12
## 5.548 4.529 3.566 2.946 2.369 2.240 1.831 1.373 1.140 1.027 0.712 0.553
## RDA13 RDA14
## 0.389 0.303
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9
## 4.965 2.582 2.059 1.740 1.446 0.930 0.743 0.548 0.458
```

```
anova(vare.rda)
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(X = varespec, Y = scale(varechem), scale = T)
##          Df Variance      F Pr(>F)
## Model    14   28.527 1.1852  0.185
## Residual  9   15.473
```

```
plot(vare.rda)
```

From the printed object, we see that 65 % of inertia is accounted for by the constraining variables. The plot shows the loadings associated with the species in the samples, and the variables in the explanatory table over the first two constrained axes (explaining 23 % of the total variation; you can calculate this yourself or find it stored in the summary object under `summary(vare.rda)[['cont']][['importance']]`). `vegan` includes an anova-like function that tests the significance of the constraints using permutation. Here we see that even though the variance explained is high, the model containing the constraint does

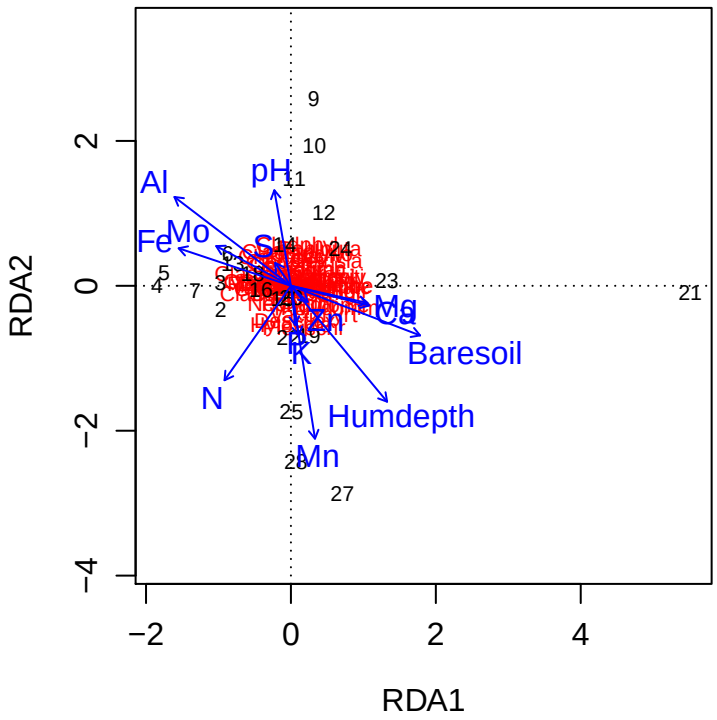

Figure 1.12: RDA of plant communities after constraining variation by soil variables.

not provide a significantly better fit. This may be due to the penalty associated with the large number of variables in the explanatory matrix; simply including all of the data without providing any thought as to which data may be useful is unlikely to generate an interpretable response.

There are a few options for selecting only a subset of variables that appear to be important predictors of changes in the community matrix. One approach is to use `envfit` to identify those variables whose vectors are significantly correlated with the site loadings.

```
vare.rda <- rda(varespec, scale(varechem), scale=T)
envfit(vare.rda, scale(varechem))
```

```
##
## ***VECTORS
##
##               RDA1     RDA2     r2 Pr(>r)
## N         -0.46296 -0.88638 0.2575  0.041 *
## P          0.05485 -0.99849 0.0320  0.734
## K          0.08770 -0.99615 0.0517  0.563
## Ca         0.95338 -0.30177 0.0980  0.337
## Mg         0.96319 -0.26883 0.0929  0.328
## S         -0.43111  0.90230 0.0142  0.873
## Al        -0.68922  0.72455 0.3562  0.014 *
## Fe        -0.91722  0.39839 0.2135  0.087 .
## Mn         0.08864 -0.99606 0.4900  0.003 **
## Zn         0.59628 -0.80278 0.0094  0.913
## Mo        -0.81301  0.58225 0.1133  0.280
## Baresoil   0.89251 -0.45103 0.2942  0.022 *
## Humdepth   0.50530 -0.86295 0.4058  0.005 **
## pH        -0.10028  0.99496 0.1921  0.111
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 999
```

```
varechem_subs <- varechem[,c('N','Al','Fe','Mn','Baresoil','Humdepth')]
vare.rda.envfit <- rda(varespec,
                       scale(varechem_subs),
                       scale=T)
anova(vare.rda.envfit)
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(X = varespec, Y = scale(varechem_subs), scale = T)
##          Df Variance      F Pr(>F)
## Model     6   14.139 1.3416  0.041 *
## Residual 17   29.861
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To test the significance of the relationship between the community and environmental matrices, we only included those variables whose vectors were significantly (or marginally-significantly) correlated with the sites loadings on the first two axes. We can see that this relationship is now significant, where it wasn't when including all environmental variables.

Try this yourself Note that the code provided here only identifies variables whose vectors are correlated with the loadings on the first two axes. This is appropriate in cases where these axes account for most of the variation in the community matrix. Use the `choices` argument in envfit to increase the number axes over which environmental vectors are correlated. See how this affects the significance of the relationship between the community and environmental matrices.

If the goal is to identify which variables are generally important, it would be better to use forward and/or reverse selection to select a subset of significant variables. We can do this using the `ordistep()` function in `vegan` after fitting two models: one containing all predictors and the other containing no predictors. The function prints output to the screen after each iteration and stops once adding/removing a variable to/from the model no longer significantly improves the model fit. Here we select potential drivers of plant community composition from a standardised matrix of environmental variables, using both forward and backward selection.

```
# select particular variables to proceed with
# here we use both forward and backward selection

# generate a new dataframe containing scaled predictor variables
# since difficult to do this in the formula
varechem.scld <- decostand(varechem, method='standardize')

# have to use the formula interface so generate a new 'rda' object
# including all predictors (use '.' after the '~')
vare.rda <- rda(varespec ~ ., data=varechem.scld, scale=T)

# set up the null case with no predictors (be sure to include the
# 'data' argument, even though no predictors)
vare.pca <- rda(varespec ~ 1, data=varechem.scld, scale=T)
```

```
# select variables in each predictor table (output not shown)
step.env <- ordistep(vare.pca, scope=formula(vare.rda))
```

We can then look at the result for when including only those variables that best explain variation between communities. We can also look at the significance level associated with each variable included in this model.

```
# show the object summary
step.env
```

```
## Call: rda(formula = varespec ~ Humdepth, data = varechem.scld,
## scale = T)
##
##                Inertia Proportion Rank
## Total         44.00000    1.00000
## Constrained    3.38495    0.07693    1
## Unconstrained 40.61505    0.92307   22
## Inertia is correlations
##
## Eigenvalues for constrained axes:
##  RDA1
## 3.385
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
## 8.373 4.535 3.749 3.245 2.960 2.737 2.181 1.954
## (Showed only 8 of all 22 unconstrained eigenvalues)
```

```
# evaluate the statistical significance of the constraint
anova(step.env)
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = varespec ~ Humdepth, data = varechem.scld, scale = T)
##          Df Variance      F Pr(>F)
## Model     1    3.385 1.8335  0.033 *
## Residual 22   40.615
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# presents results in an ANOVA-like table with the retained predictors
step.env$anova
```

```
##            Df    AIC      F Pr(>F)
## + Humdepth  1 91.878 1.8335   0.03 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Try this yourself Use the `direction` argument to perform either 'forward' or 'backward' selection of predictor variables to include. Does this result in different variables being selected?

Variations to RDA include transformation of the response matrix prior to analysis (tb-RDA) using `decostand` (as in Section 1.3.2) or generating a response matrix from principle coordinates after PCoA (distance based [db]-RDA or constrained analysis of principal coordinates [CAP]; as in Section 1.3.3). We'll deal

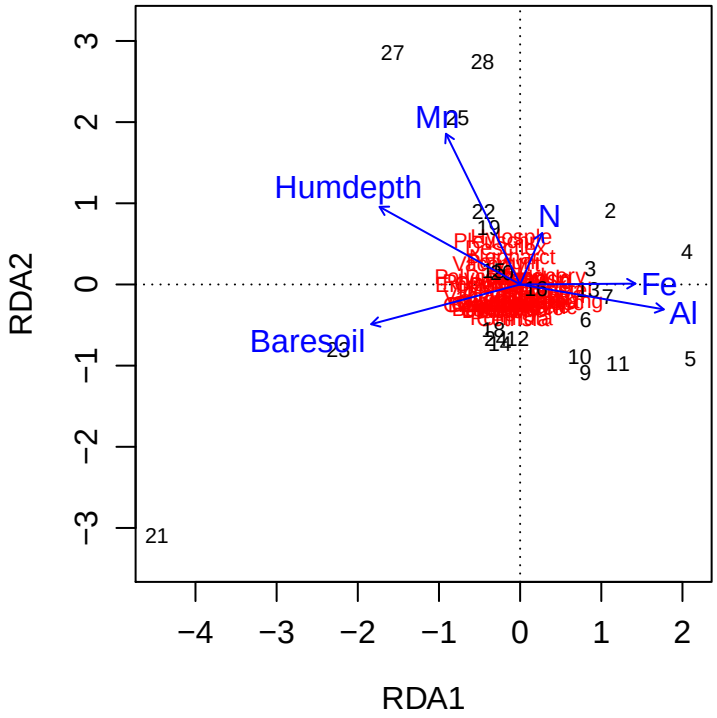

Figure 1.13: RDA ordination of sites based on variation in plant communities, constrained by significant soil properties

with these in subsequent sections.

1.4.2.2 Canonical correspondence analysis (CCA)

CCA is an extension of CA, but serves the same purpose as RDA in relation to PCA. CCA is performed using `cca` in `vegan`. Its use in `vegan` is very similar to `rda()`, so we will not deal with it further here. It may be preferred over RDA for long gradients, over which species distributions are limited to a subset of the gradient and do not overlap for many species (recall Section 1.3.2.1), but db-RDA or NMDS-based approaches can also be used.

1.4.2.3 More manipulating graphics from *vegan* objects

The plots that are produced from using `plot` on `vegan` objects are useful for visualising patterns but not so nice for publication. For example, look at the result for the analysis performed in Section 1.4.2.1, shown in Fig. 1.13.

We can make much nicer plots (Fig. 1.14) showing the loadings associated with species and soil properties by extracting relevant information from the resulting object.

```
## get the relevant information from the RDA object

# 'sites' and 'species' loadings can be retrieved using 'scores' function
sites <- scores(vare.rda.envfit, display='sites')
spp <- scores(vare.rda.envfit, display='species')

# proportion of variation explained by each constrained axis (for plot axes)
```

```
# can't just use sum(eig) for denominator, this vector only includes constraint
eig <- vare.rda.envfit[['CCA']][['eig']]
eig / vare.rda.envfit[['tot.chi']]
```

```
##       RDA1       RDA2       RDA3       RDA4       RDA5       RDA6
## 0.10188359 0.08724602 0.06177211 0.03575530 0.02230310 0.01238436
```

```
## plot one figure on top of another
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))
## plot showing loadings for species (response variables)
plot(sites, pch=16, col='grey', xlim=c(-5, 5), ylim=c(-4, 4), main='Species',
         xlab='Axis 1 (10.2 %)', ylab='Axis 2 (8.7 %)')
# add lines to separate the plot regions
abline(v=0, h=0, lty='dashed')
# use a multiplier (here, 7) to spread the points away from the origin for readability
# also, only print the first four letters to reduce overlap
text(spp[, 1] * 7, spp[, 2] * 7, substr(rownames(spp), 1, 4), col='blue', cex=0.6)

## plot showing loadings for soil properties (constraining variables)
plot(sites, pch=16, col='grey', xlim=c(-5, 5), ylim=c(-4, 4), main='Constraints',
         xlab='Axis 1 (10.2 %)', ylab='Axis 2 (8.7 %)')
# add lines to separate the plot regions
abline(v=0, h=0, lty='dashed')
# add text and arrows linked to variable labels
text(vare.rda.envfit, display='bp', col='blue', cex=0.75)
```

Try this yourself In the above example, adjust the `par` settings to modify the plot to your liking.

1.4.3 More than two tables - variation partitioning

Canonical approaches calculate the correspondence between community composition and environmental properties, and allow for the estimation of each variable's explanatory power as a predictor of community shifts. These environmental properties may belong to different categories (e.g., chemical, physical, climatic) and/or may be measured along known spatial gradients. In these cases, we may want to partition variation in community composition to groups of variables in order to gain a general sense of how important each group of variables is for driving community shifts. We can do this using the `varpart` function in the `vegan` package.

In this example, we partition variation in the `varespec` plant data to two categories of environmental data: soil elemental chemistry and soil exposure (prevalence of bare soil and depth of the humus layer). We perform variation partitioning by including both types of predictor matrices as separate arguments, and only including the variables that appeared to be important for explaining variation (using the `envfit` results from above in this example).

```
# partition variation among two predictor tables:
#   1) soil elemental chemistry ('N', 'Al', 'Fe', and 'Mn')
#   2) soil exposure ('Baresoil' and 'Humdepth')
vare.var <- varpart(varespec,
                                        ~ N + Al + Fe + Mn,
                                        ~ Baresoil + Humdepth,
                                        data=varechem.scld, scale=T)
```

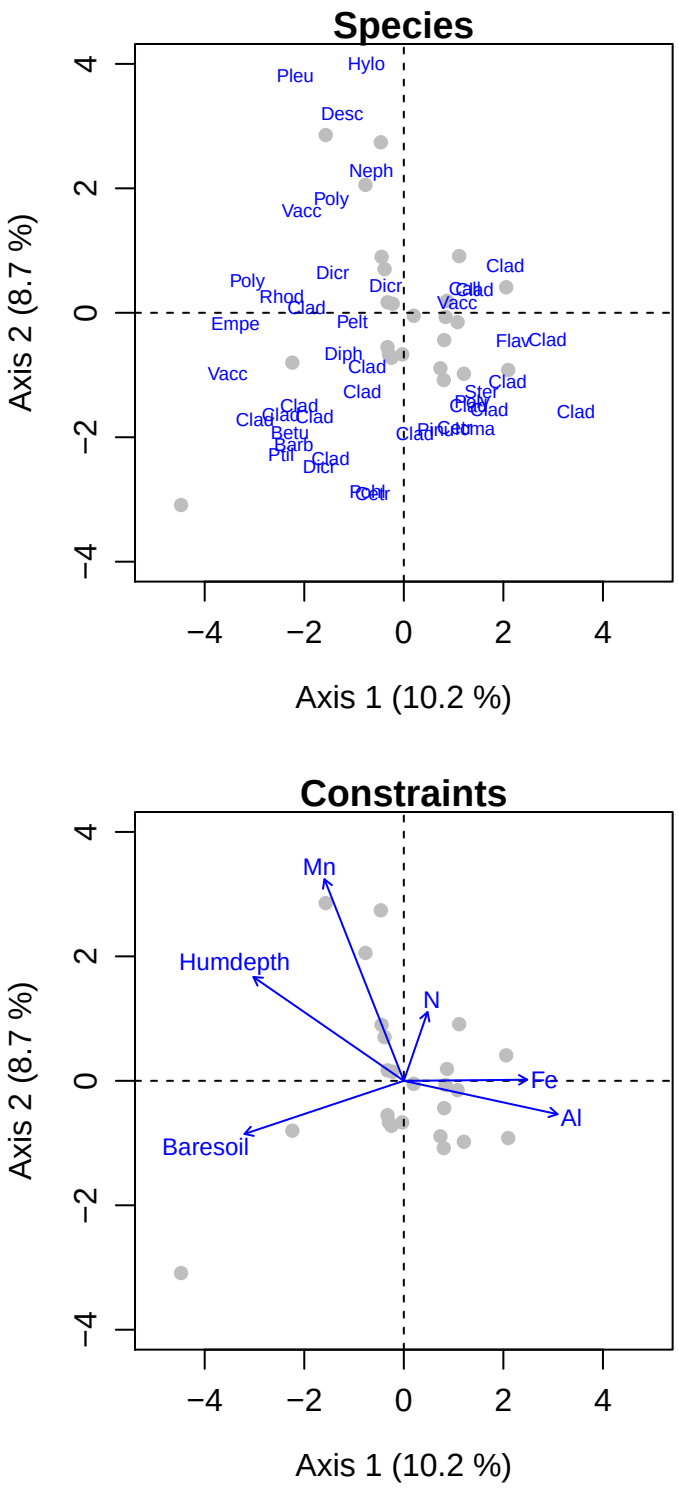

Figure 1.14: RDA ordination of sites based on variation in plant communities, showing plant species loadings (top) and loadings for constraining variables (bottom)

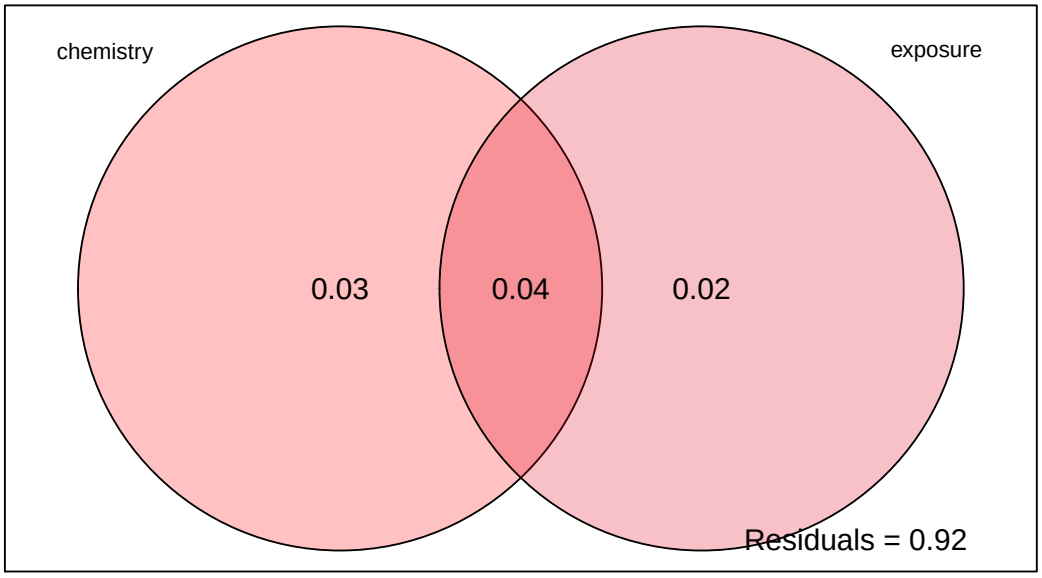

Figure 1.15: Venn diagram showing variation partitioned to variables associated individually with soil chemistry, with soil exposure, or across both.

```
# plot variation association with each partition ('bg' used to set colours)
plot(vare.var, bg=1:3, Xnames=c('chemistry', 'exposure'), id.size=0.75)
```

```
# show variation associated with each partition and across both partitions
vare.var
```

```
##
## Partition of variance in RDA
##
## Call: varpart(Y = varespec, X = ~N + Al + Fe + Mn, ~Baresoil +
## Humdepth, data = varechem.scld, scale = T)
## Columns of Y were scaled to unit variance
##
## Explanatory tables:
## X1:  ~N + Al + Fe + Mn
## X2:  ~Baresoil + Humdepth
##
## No. of explanatory tables: 2
## Total variation (SS): 1012
##              Variance: 44
## No. of observations: 24
##
## Partition table:
```

```
##                        Df R.squared Adj.R.squared Testable
## [a+b] = X1              4   0.22774       0.06516     TRUE
## [b+c] = X2              2   0.13468       0.05227     TRUE
## [a+b+c] = X1+X2         6   0.32134       0.08182     TRUE
## Individual fractions
## [a] = X1|X2             4                 0.02955     TRUE
## [b]                     0                 0.03561    FALSE
## [c] = X2|X1             2                 0.01666     TRUE
## [d] = Residuals                           0.91818    FALSE
## ---
## Use function 'rda' to test significance of fractions of interest
```

The partition table displayed above has two sections. The first contains the total amount of variation associated with each partition and across both partitions. The second section of the partition table looks at the individual fractions, or the amount of variation that is attributed solely to each partition and the amount that cannot be attributed to an individual partition. The Venn diagram in Figure 1.15 summarises the results in the second section.

The statistical significance of the values in the first section of the partition table can be tested by using the `anova()` function on an `rda()` object.

```
# test significance of variation in partition 'X1' (chemistry)
anova(rda(varespec ~ N + Al + Fe + Mn, data=varechem.scld, scale=T))
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = varespec ~ N + Al + Fe + Mn, data = varechem.scld, scale = T)
##          Df Variance      F Pr(>F)
## Model     4   10.021 1.4008  0.028 *
## Residual 19   33.979
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# test significance of variation in partition 'X2' (exposure)
anova(rda(varespec ~ Baresoil + Humdepth, data=varechem.scld, scale=T))
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = varespec ~ Baresoil + Humdepth, data = varechem.scld, scale = T)
##          Df Variance      F Pr(>F)
## Model     2    5.926 1.6343  0.005 **
## Residual 21   38.074
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# test significance of variation in both partitions
anova(rda(varespec ~ N + Al + Fe + Mn + Baresoil + Humdepth,
          data=varechem.scld, scale=T))
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
```

```
## Model: rda(formula = varespec ~ N + Al + Fe + Mn + Baresoil + Humdepth, data = varechem.scld, scale =
##          Df Variance      F Pr(>F)
## Model     6   14.139 1.3416  0.026 *
## Residual 17   29.861
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

These are all significant, which is not surprising because we already demonstrated this in the section 1.4.2.1. The statistical significance of the individual fractions in the second section of the partition table can also be tested by using the `anova()` function on an `rda()` object that uses the `Condition()` function to remove variation associated with variables in the other partition.

```
# test significance of variation in partition 'X1' (chemistry) after
# accounting for variation in 'X2'
anova(rda(varespec ~ N + Al + Fe + Mn
          + Condition(Baresoil + Humdepth), data=varechem.scld, scale=T))
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = varespec ~ N + Al + Fe + Mn + Condition(Baresoil + Humdepth), data = varechem.sc
##          Df Variance     F Pr(>F)
## Model     4   8.2132 1.169  0.187
## Residual 17  29.8608
```

```
# test significance of variation in partition 'X2' (exposure) after
# accounting for variation in 'X1'
anova(rda(varespec ~ Baresoil + Humdepth
          + Condition(N + Al + Fe + Mn), data=varechem.scld, scale=T))
```

```
## Permutation test for rda under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = varespec ~ Baresoil + Humdepth + Condition(N + Al + Fe + Mn), data = varechem.sc
##          Df Variance      F Pr(>F)
## Model     2   4.1185 1.1724  0.233
## Residual 17  29.8608
```

Neither of the individual fractions, associated only with the 'exposure' partition or the 'chemistry' partition, is significant. This is, again, not surprising – the variation explained by each of these individual partitions is small and there is a larger fraction of variation that is explained but cannot be partitioned individually (the overlapping section of the Venn diagram in 1.15).

1.4.3.1 Variation Partitioning - incorporating spatial processes

In some cases, we have information on the physical locations from which our samples were collected. We can include these spatial characteristics in our analyses because the patterns linked to these variables can represent processes associated with unmeasured (and spatially autocorrelated) environmental variation and/or species dispersal among each of the locations. To do this, we use the `pcnm()` function in the `vegan` library. This function calculates Principal Coordinates of Neighbour Matrices (PCNMs) to generate a dataframe containing variables that represent different spatial scales. We demonstrate this here using the 'mite' data that come with `vegan`, including count data for 35 mite species observed across 70 samples.

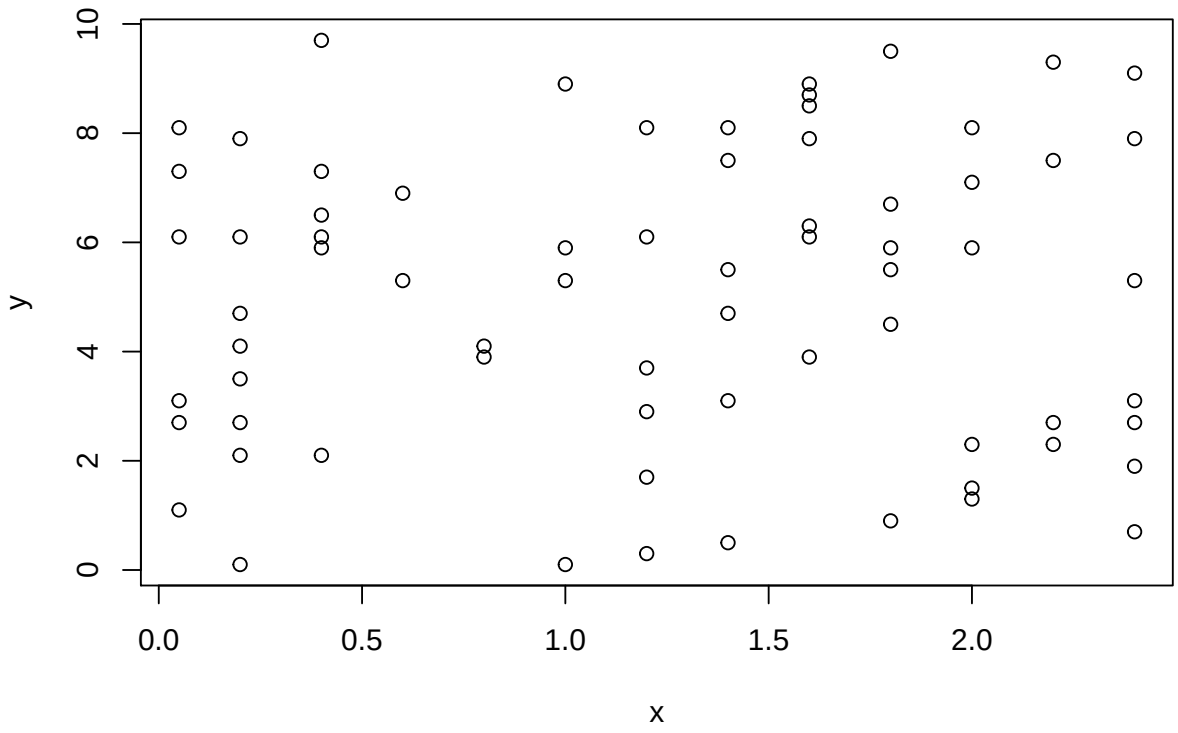

Figure 1.16: Spatial locations of mite samples, along two dimensions (x, y).

```
# load tables containing species, environmental variables, and geographic coordinates

# species-sample table - species in columns, samples in rows
data(mite)
dim (mite)
```

```
## [1] 70 35
```

```
# five environmental variables associated with each sample location
data(mite.env)
summary(mite.env)
```

```
##     SubsDens        WatrCont         Substrate   Shrub         Topo
##  Min.   :21.17   Min.   :134.1   Sphagn1  :25   None:19   Blanket:44
##  1st Qu.:30.01   1st Qu.:314.1   Sphagn2  :11   Few :26   Hummock:26
##  Median :36.38   Median :398.5   Sphagn3  : 1   Many:25
##  Mean   :39.28   Mean   :410.6   Sphagn4  : 2
##  3rd Qu.:46.81   3rd Qu.:492.8   Litter   : 2
##  Max.   :80.59   Max.   :827.0   Barepeat : 2
##                                  Interface:27
```

```
# 'x' and 'y' coordinates associated with each sample location
data(mite.xy)
plot(mite.xy)
```

```
# calculate PCNMs from a Euclidean distance matrix of sample coordinates
# and extract scores associated with these new variables
# convert to data.frame for downstream steps
mite.pcnm <- as.data.frame(scores(pcnm(dist(mite.xy))))
dim(mite.pcnm)
```

```
## [1] 70 43
```

The patterns represented by the variables in the resulting `pcnm` object are fairly straightforward when samples are evenly spaced in one direction (see Borcard and Legendre, 2002, Ecological Modelling 153:51âĂŞ68). Two-dimensional sampling, especially when samples are not evenly spaced, results in variables with patterns that are not as easy to interpret, but the first axes tend to represent large scale patterns while the later axes tend to represent smaller scale patterns. In figure 1.17 we use colour to plot the loadings on the first six PCNM axes to visualise the spatial patterns that they represent.

```
# set up a multipanel graphics window
par(mfrow=c(2, 3))

# set colour palette with ten levels along a gradient from red to blue
# from Chapter 4
blueredfun <- colorRampPalette(c("blue","red"))
palette(blueredfun(10))

# for each of the first six PCNM axes, use colour to represent loadings
plot(mite.xy, pch=16, col=cut(mite.pcnm[[1]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
plot(mite.xy, pch=16, col=cut(mite.pcnm[[2]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
plot(mite.xy, pch=16, col=cut(mite.pcnm[[3]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
plot(mite.xy, pch=16, col=cut(mite.pcnm[[4]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
plot(mite.xy, pch=16, col=cut(mite.pcnm[[5]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
plot(mite.xy, pch=16, col=cut(mite.pcnm[[6]], breaks=10), cex.lab=1.5, cex.axis=1.3, cex=2)
```

Now that we have two matrices, one representing variation in measured environmental variables (`mite.env`) and the other representing spatial distributions of samples (`mite.pcnm`), we can estimate the amount of variation in community composition that each explains (shown in Figure 1.18).

```
# do predictor matrices explain community composition, and how much?

# partition variation among three predictor tables:
#   1) substrate ('Substrate', 'SubsDens', and 'WatrCont')
#   2) landscape, i.e., shrub density adn microtopography ('Shrub' and 'Topo')
#   3) space ('mite.pcnm')
mite.var <- varpart(mite,
                                        ~ Substrate + SubsDens + WatrCont,
                                        ~ Shrub + Topo,
                                        mite.pcnm, data=mite.env)
plot(mite.var, bg=1:3, Xnames=c('substrate', 'landscape', 'space'), id.size=0.75)
```

The individual fraction associated with 'landscape' is missing because this number is negative. These numbers represent R^2 values after adjusting for the number of explanatory variables in each partition ('adjusted R^2') and will be negative when the raw R^2 is very small

Try this yourself In the above example, use `ordistep()` to select significant PCNM axes and repeat the variation partitioning. Then, use `rda()` to evaluate the signficance of variation explained by each of the individual partitions.

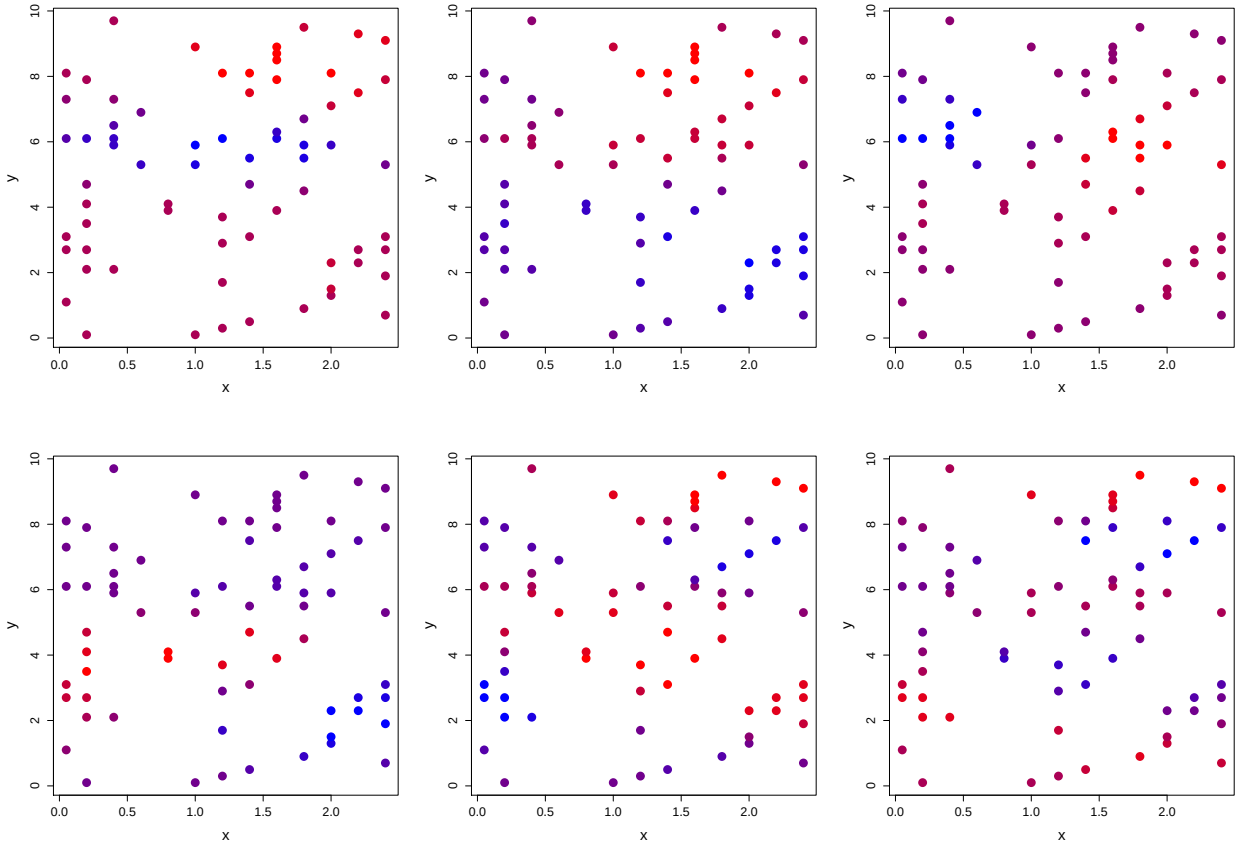

Figure 1.17: Loadings associated with six PCNM axes, plotted against the geographic position of where each sample was collected. Loadings are scaled from positive (red) to negative (blue).

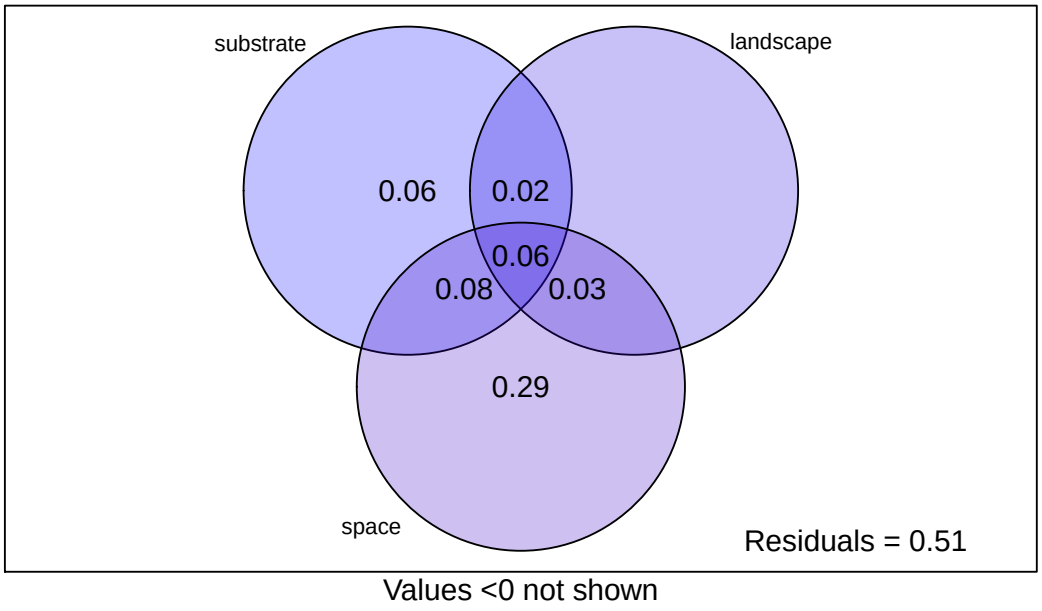

Figure 1.18: Venn diagram showing variation partitioned to variables associated individually with soil chemistry, with soil exposure, with PCNM axes, or across multiple partitions.

1.4.4 Using categorical variables in canonical analyses

In the variation partitioning example above, we used the species-sample matrix as our response matrix (essentially redundancy analysis), not a distance matrix (principal coordinates analysis) even though our responses are counts. This is because there is a debate in the literature about whether it is appropriate to use a distance-based approach for variation partionting, with the answer tending towards 'no' (see Laliberté, 2008, Ecology 89:3232âĂŞ3237 and papers cited within). We can perform principal coordinates analysis using the `capscale` function when visualising patterns in the response and relationships with the explanatory variables and evaluating the signficance of particular explanatory variables.

```
# use decostand to perform 'Hellinger' transformation of mite data
mite.hel <- decostand(mite, method='hellinger')

# set up full and null models for 'ordistep'
mite.cap1 <- capscale(mite.hel ~ ., data=mite.env, dist='bray')
mite.cap0 <- capscale(mite.hel ~ 1, data=mite.env, dist='bray')
```

```
# perform forward and backward selection of explanatory variables
# output not shown
step.env <- ordistep(mite.cap0, scope=formula(mite.cap1))
```

```
# look at the significant variables
step.env$anova
```

```
##              Df    AIC       F Pr(>F)
## + WatrCont    1 151.62 26.2082  0.005 **
## + SubsDens    1 147.61  6.0122  0.005 **
## + Topo        1 142.31  7.2483  0.005 **
## + Shrub       2 141.48  2.2881  0.005 **
## + Substrate   6 142.65  1.6169  0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# view ordination
plot(step.env)
```

From the ANOVA table, we can see that all of the variables are significant, but we cannot see which levels are important for the factor variables. We can kind of see this from the ordination plot (the levels that are furthest from the centre are most likely to be significant), but it would be useful to do a formal statistical test for each factor level. We can use `ordistep` to do this after transforming the factor levels each into an individual 'dummy' variable. For this example, we use the `dudi.mix` function in the `ade4` library to do this.

```
# load the 'ade4' library
# note the warning due to multiple packages having a function called 'cca'!
library('ade4')

# transform the factor variables into individual dummy variables
# modified variables stored in the 'tab' element of the resulting list object
# use 'scannf' and 'nf' arguments to suppress dialogue
mite.env.mod <- dudi.mix(mite.env, scannf=F, nf=2)$tab

# set up full and null models for 'ordistep'
mite.cap1 <- capscale(mite.hel ~ ., data=mite.env.mod, dist='bray')
mite.cap0 <- capscale(mite.hel ~ 1, data=mite.env.mod, dist='bray')
```

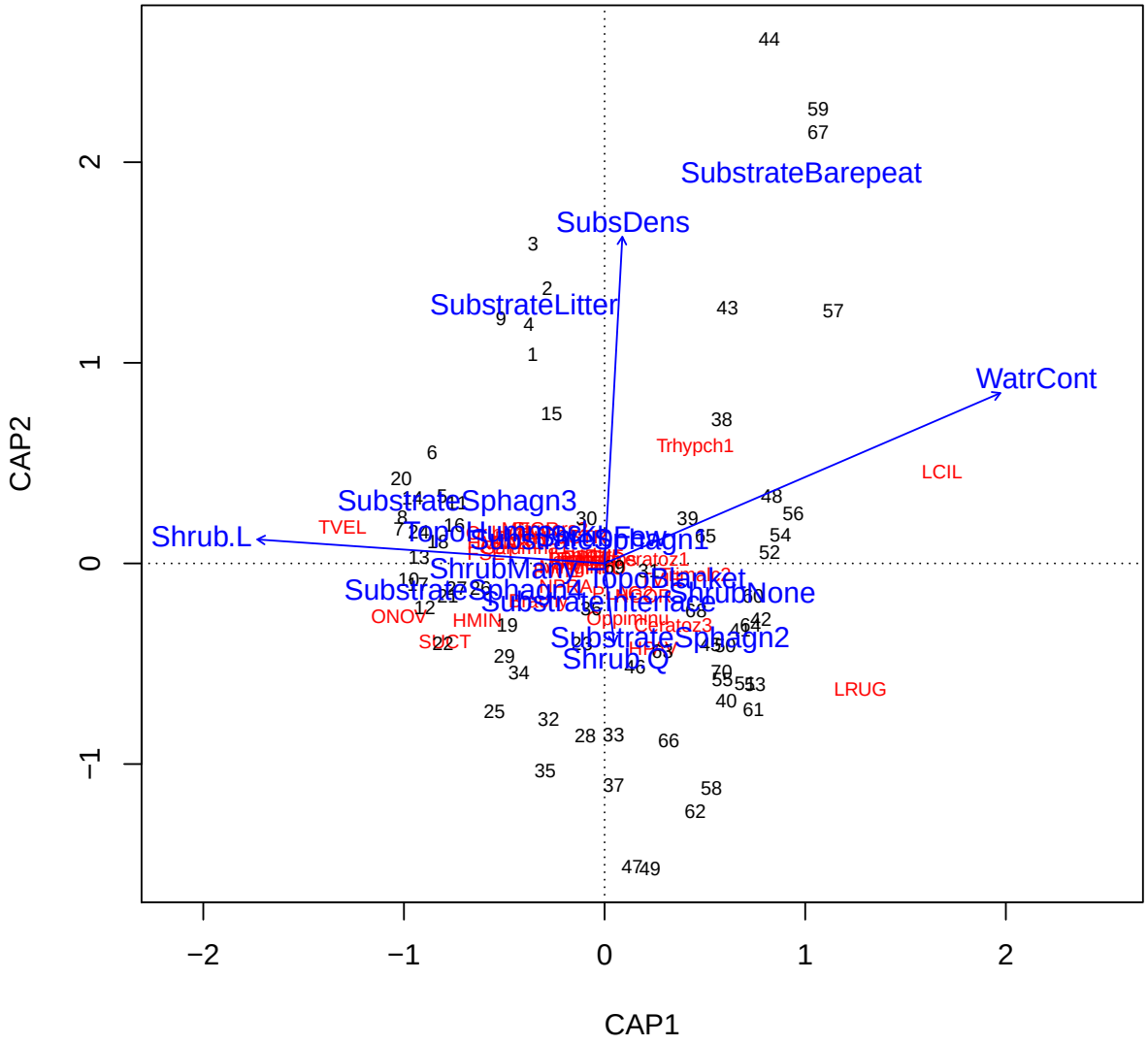

Figure 1.19: Ordination of mite data based on CAP analysis including continuous and categorical variables.

```
# perform forward and backward selection of explanatory variables
# (output not shown)
step.env <- ordistep(mite.cap0, scope=formula(mite.cap1))
```

```
# look at the significant variables
step.env$anova
```

```
##                   Df    AIC       F Pr(>F)
## + WatrCont         1 151.62 26.2082  0.005 **
## + SubsDens         1 147.61  6.0122  0.005 **
## + Topo.Blanket     1 142.31  7.2483  0.005 **
## + Shrub.L          1 142.09  2.1008  0.010 **
## + Subst.Barepeat   1 141.15  2.7422  0.005 **
## + Subst.Sphagn1    1 140.62  2.3193  0.010 **
## + Shrub.Q          1 139.78  2.5700  0.010 **
## + Subst.Sphagn3    1 140.10  1.4735  0.045 *
## - Subst.Sphagn3    1 139.78  1.4735  0.140
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# view ordination
plot(step.env)
```

The nonsignficant explanatory variables not plotted, making it easier to see relationships among significant variables, samples, and mite species.

1.4.4.1 Indicator species analysis

We can visualise relationships between species and environmental variables by looking at ordinations, but it may be useful to identify species that are significantly associated with particular environmental variables. This can be done through Indicator Species analysis using functions in the `labdsv` library or the `indicspecies` library. Below we show examples using the `indval` function from the `labdsv` library.

```
# load library
library(labdsv)

## example using a factor variable

# calculate indicator values for each species
ind.topo <- indval(mite, mite.env$Topo)

# calculate adjusted P-values for each species and show significant species
topo.pval <- p.adjust(ind.topo$pval, method='bonferroni')
topo.pval[topo.pval < 0.05]
```

```
##     RARD     TVEL     ONOV     SUCT Oribatl1 Galumna1     FSET     LRUG
##    0.035    0.035    0.035    0.035    0.035    0.035    0.035    0.035
```

```
# show indicator values for significant indicator species in each group
topo.ind <- ind.topo[['indval']]
topo.ind[rownames(topo.ind) %in% names(topo.pval[topo.pval < 0.05]), ]
```

```
##             Blanket    Hummock
## RARD     0.01369430 0.54122471
## TVEL     0.07548822 0.71867805
## ONOV     0.22046222 0.73782871
```

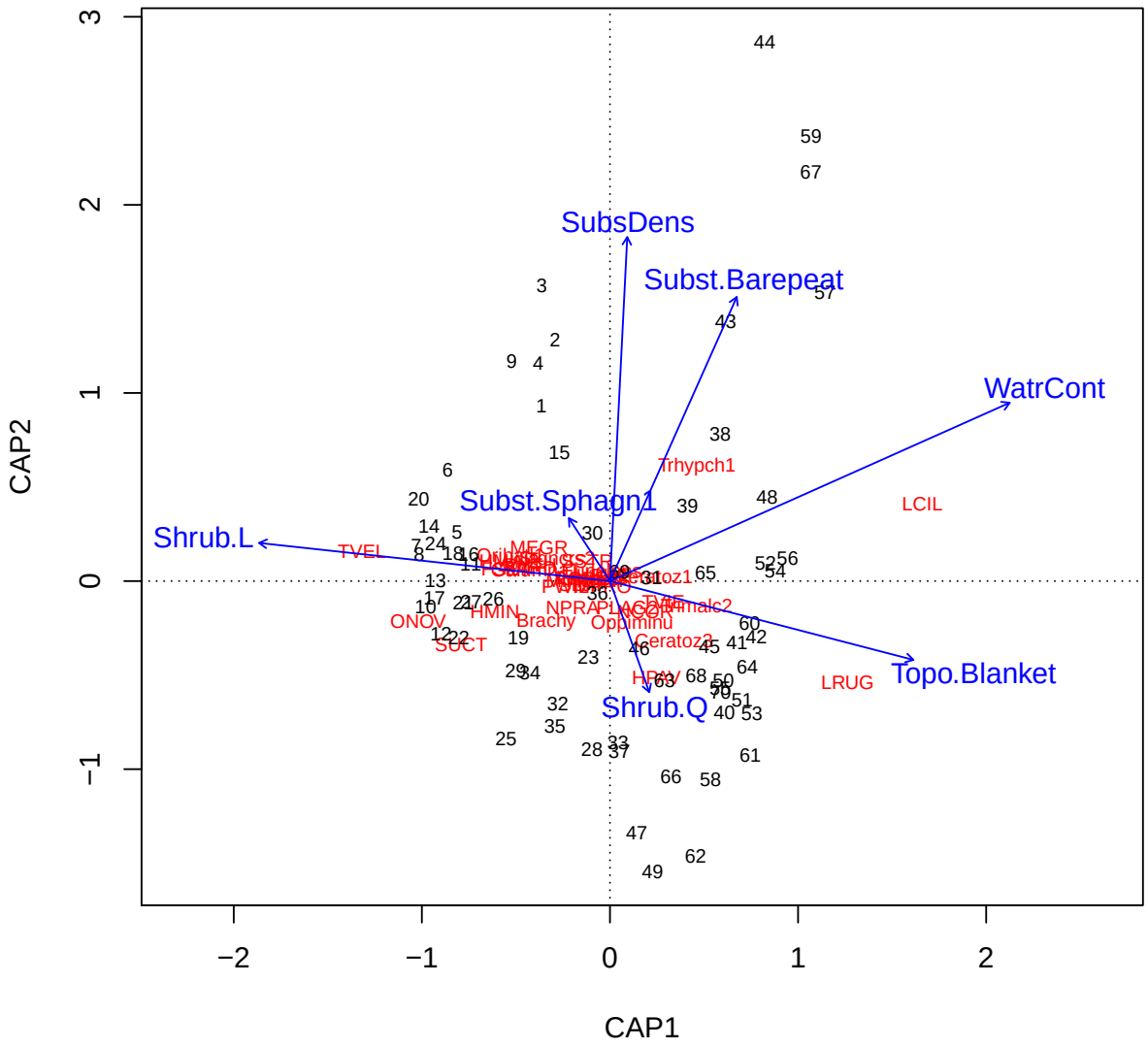

Figure 1.20: Ordination of mite data based on CAP analysis including continuous and categorical variables after removal of nonsignificant levels in categorical variables.

```
## SUCT     0.29871725 0.67942539
## Oribatl1 0.06621842 0.53720770
## Galumna1 0.03379908 0.54522598
## FSET     0.04719869 0.57249563
## LRUG     0.79702367 0.04267176
```

```
# show frequencies in each group for significant indicator species
topo.frq <- ind.topo[['relfrq']]
topo.frq[rownames(topo.frq) %in% names(topo.pval[topo.pval < 0.05]), ]
```

```
##            Blanket   Hummock
## RARD     0.1136364 0.6153846
## TVEL     0.3409091 0.9230769
## ONOV     0.8409091 1.0000000
## SUCT     0.9318182 1.0000000
## Oribatl1 0.2500000 0.7307692
## Galumna1 0.1590909 0.6923077
## FSET     0.2727273 0.6923077
## LRUG     0.9090909 0.3461538
```

```
## example using a continuous variable

# create grouping categories from the continuous variable
groups <- cut(mite.env$WatrCont, breaks=3)

# calculate indicator values for each species
ind.dens <- indval(mite, groups)

# calculate adjusted P-values for each species and show significant species
dens.pval <- p.adjust(ind.dens$pval, method='bonferroni')
dens.pval[dens.pval < 0.05]
```

```
##  TVEL  ONOV  SUCT
## 0.035 0.035 0.035
```

```
# can use code from the first example to interpret group membership for indicator species
```

1.4.5 'Experimental' frameworks: more working with factors

1.4.5.1 Cluster analysis

We often collect multivariate data in the context of experimental studies or in the context of observed categorical explanatory variables that are of interest (as seen in section 1.4.4). We could perform ANOVA for each variable individually, but this amplifies the potential for Type I error and does not account for potential collinearity among response variables. In addition, we may not be interested in particular response variables, but in how multiple variables are responding as a whole. In this example, we use the Tibetan Plateau plant community data from Section **??**. We will use the complete-linkage algorithm, which is the default and aims to maximise the distances among clusters, but other algorithms are available and may be more appropriate.

```
tibplat <- read.csv('tibplat.csv')

# Species data are in columns 3:10
tib.clust <- hclust(vegdist(tibplat[,3:10],'bray'))
plot(tib.clust)
```

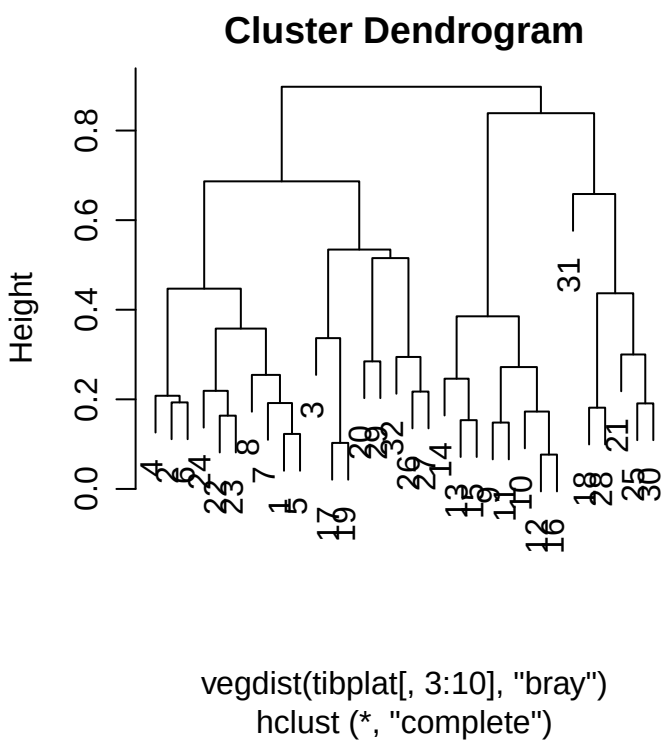

Figure 1.21: Hierarchical clustering of Tibetan Plateau plant communities based on Bray-Curtis dissimilarities.

The dendrogram in Fig. 1.21 indicates that the plant communities appear to be hierarchically clustered into groups. The label on the tips represent the rownames in the dataframe but are not very informative on their own. Here it is useful to provide a more informative vector with which to label the tips. We will use the information on the management practices associated with each plant community (Fig. 1.22).

```
tib.clust <- hclust(vegdist(tibplat[,3:10],'bray'))
plot(tib.clust, labels=tibplat[['fertilization']], main='fertilization')
plot(tib.clust, labels=tibplat[['enclosure']], main='enclosure')
```

Groups do appear to be linked to fertilization and grazing, but there are a few examples in which a plant community of one management type is quite divergent from others of that type. We need a statistical tool to test the hypothesis that plant communities within each management type are more similar to each other than to those under other forms of management.

Analysis of similarities (ANOSIM) is one approach to test this hypothesis, using the `anosim()` function in the `vegan` package. This is a nonparametric test; distances are transformed into ranks and the mean of the ranks for distances classified as within the groups is compared to that for the between-group distance ranks. However, there are concerns about how ANOSIM results should be interpreted (also mentioned on the help page for the function) and PerMANOVA (Permutational ANOVA) is usually preferred.

PerMANOVA is potentially more powerful than ANOSIM because the distances are preserved and not rank-transformed; the approach partitions variance among and within groups through calculations of sums of squared distances and calculates pseudo F-statistics. In addition, it is possible to estimate the effects associated with multiple factors and their interactions. The function to perform PerMANOVA is `adonis` within `vegan` and the result is interpreted in the same way as an ANOVA table.

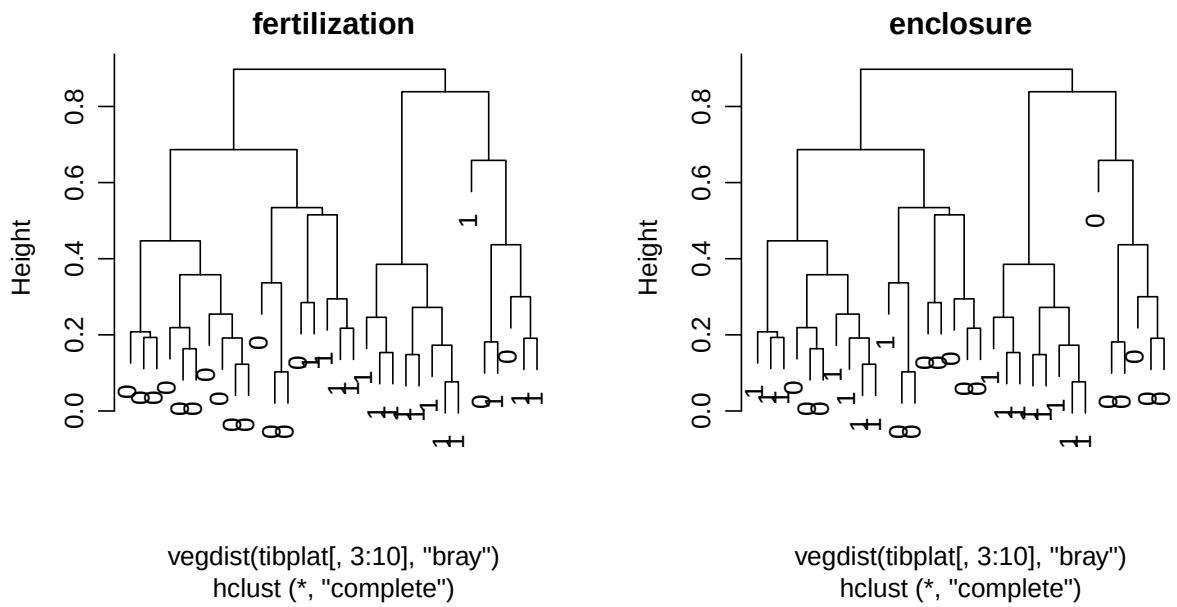

Figure 1.22: Hierarchical clustering of Tibetan Plateau plant communities based on Bray-Curtis dissimilarities, with tips labelled by management system (fertilization, enclosed).

```
adonis(vegdist(tibplat[,3:10]) ~ fertilization*enclosure, data=tibplat)

##
## Call:
## adonis(formula = vegdist(tibplat[, 3:10]) ~ fertilization * enclosure,      data = tibplat)
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##                          Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## fertilization             1    1.2320 1.23202  20.757 0.25944  0.001 ***
## enclosure                 1    1.1358 1.13583  19.137 0.23918  0.001 ***
## fertilization:enclosure   1    0.7190 0.71901  12.114 0.15141  0.001 ***
## Residuals                28    1.6619 0.05935         0.34997
## Total                    31    4.7488                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.4.5.2 More manipulating graphics from *vegan* objects

Let's produce a figure that visualises the relationship between the experimental treatments and the results of the PCoA used in the PERMANOVA in the previous section.

```
# perform PCoA and store results in object
tib.pco <- wcmdscale(vegdist(tibplat[,3:10]), eig=T)

# return the proportions of variance explained
tib.pco$eig[tib.pco$eig >= 0] / sum(tib.pco$eig[tib.pco$eig >= 0])
```

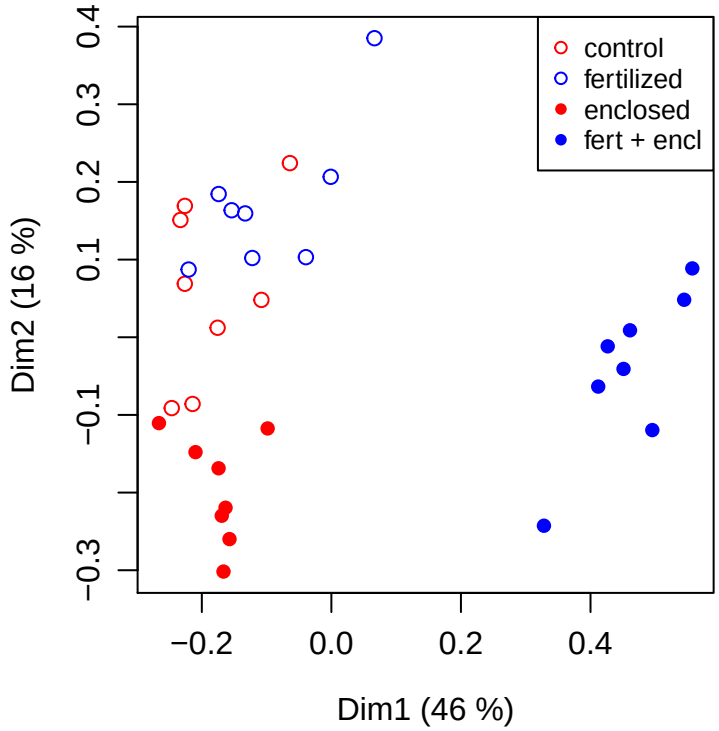

Figure 1.23: MDS ordination of sites based on variation in plant communities

```
##  [1] 0.4569200850 0.1601330893 0.1144687053 0.0828594613 0.0520696801
##  [6] 0.0447359716 0.0256728499 0.0198370466 0.0136060804 0.0091392099
## [11] 0.0067712217 0.0057549299 0.0038943360 0.0027786236 0.0011320982
## [16] 0.0002266113
```

```
palette(c('red','blue'))  # set up the colour palette
with(tibplat, plot(scores(tib.pco, display='sites'),  # identifies coordinates
                   pch=c(1,16)[enclosure], col=fertilization,   # assign symbols, colours
                   xlab='Dim1 (46 %)', ylab='Dim2 (16 %)'))  # change axis labels
legend('topright',  # position the legend
       legend=c('control', 'fertilized', 'enclosed', 'fert + encl'),  # set legend text
       pch=c(1,1,16,16), col=c('red','blue','red','blue'),  # assign symbols and colours
       cex=0.8)  # slightly reduce the size of the text and points
```

Try this yourself In the above example, adjust the `par` settings to modify the plot to your liking.

Patterns among treatments may be difficult to see in the ordination in some cases. The `vegan` package includes functions to include shapes that delimit boundaries around groups of points (the `ordihull` function) or a certain distance from the centroid (the `ordiellipse` function). Another `vegan` function (`ordispider`) connects each point to the group centroid with lines.

```
# perform PCoA and store results in object
tib.pco <- wcmdscale(vegdist(tibplat[,3:10]), eig=T)

# return the proportions of variance explained
tib.pco$eig[tib.pco$eig >= 0] / sum(tib.pco$eig[tib.pco$eig >= 0])
```

```
##  [1] 0.4569200850 0.1601330893 0.1144687053 0.0828594613 0.0520696801
##  [6] 0.0447359716 0.0256728499 0.0198370466 0.0136060804 0.0091392099
## [11] 0.0067712217 0.0057549299 0.0038943360 0.0027786236 0.0011320982
## [16] 0.0002266113
```

```
# set plot options
par(mfrow=c(3, 1))  # set plot layout
palette(c('red','blue'))  # set up the colour palette

# create a vector containing all treatment combinations
trts <- with(tibplat, interaction(fertilization, enclosure, sep='-'))
levels(trts)
```

```
## [1] "0-0" "1-0" "0-1" "1-1"
```

```
# draw a polygon around all points in a treatment
with(tibplat, plot(scores(tib.pco, display='sites'),  # identifies coordinates
                   pch=c(1,16)[enclosure], col=fertilization,   # assign symbols, colours
                   xlab='Dim1 (46 %)', ylab='Dim2 (16 %)',
                   main='ordihull'))  # change axis labels
legend('topright',  # position the legend
       legend=c('control', 'fertilized', 'enclosed', 'fert + encl'),  # set legend text
       pch=c(1,1,16,16), col=c('red','blue','red','blue'),  # assign symbols and colours
       cex=0.8)  # slightly reduce the size of the text and points
ordihull(tib.pco, trts, draw='polygon',   # draw polygon
         col=c('red','blue','red','blue'), lty=c('dashed', 'dashed', 'solid', 'solid'))

# draw an ellipse representing the 95% confidence interval from the centroid
with(tibplat, plot(scores(tib.pco, display='sites'),  # identifies coordinates
                   pch=c(1,16)[enclosure], col=fertilization,   # assign symbols, colours
                   xlab='Dim1 (46 %)', ylab='Dim2 (16 %)',
                   main='ordiellipse'))  # change axis labels
legend('topright',  # position the legend
       legend=c('control', 'fertilized', 'enclosed', 'fert + encl'),  # set legend text
       pch=c(1,1,16,16), col=c('red','blue','red','blue'),  # assign symbols and colours
       cex=0.8)  # slightly reduce the size of the text and points
ordiellipse(tib.pco, trts, kind='se', conf=0.95,   # draw ellipse
         col=c('red','blue','red','blue'), lty=c('dashed', 'dashed', 'solid', 'solid'))

# draw a spider graph linking each point to the group centroid
with(tibplat, plot(scores(tib.pco, display='sites'),  # identifies coordinates
                   pch=c(1,16)[enclosure], col=fertilization,   # assign symbols, colours
                   xlab='Dim1 (46 %)', ylab='Dim2 (16 %)',
                   main='ordispider'))  # change axis labels
legend('topright',  # position the legend
       legend=c('control', 'fertilized', 'enclosed', 'fert + encl'),  # set legend text
       pch=c(1,1,16,16), col=c('red','blue','red','blue'),  # assign symbols and colours
       cex=0.8)  # slightly reduce the size of the text and points
ordispider(tib.pco, trts, spiders='centroid',   # draw spiders
         col=c('red','blue','red','blue'), lty=c('dashed', 'dashed', 'solid', 'solid'))
```

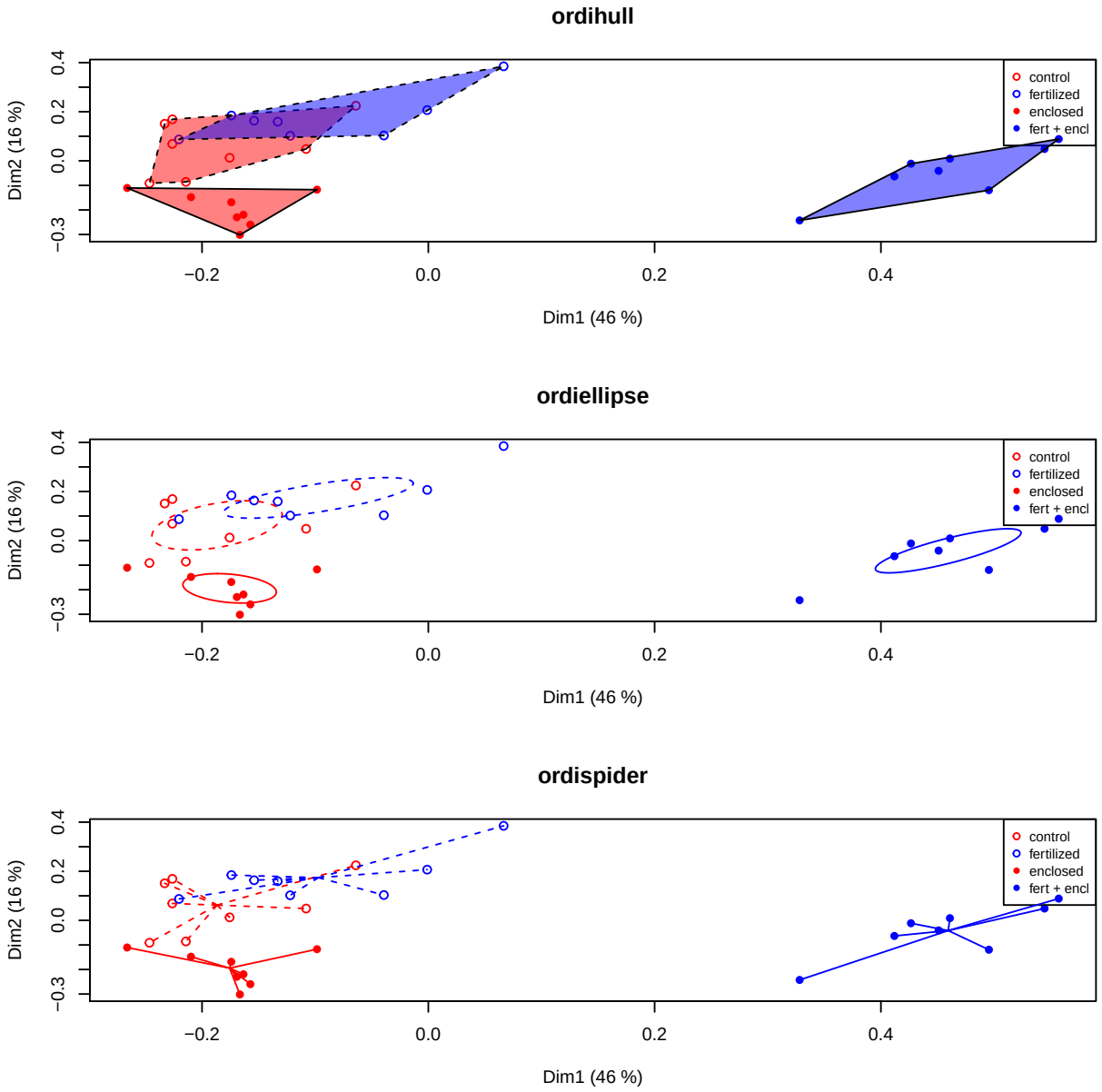

Figure 1.24: MDS ordination of sites based on variation in plant communities, further highlighting treatment groupings.

1.4.5.3 Model-based approaches

Up to this point, we have used distance-based approaches to detect patterns and to make inferences about the drivers of these patterns. 'Distance-based' essentially means that the data are transformed prior to analysis so that the similarities and dissimilarities among each of the observations can be represented by their distances in multivariate space. These approaches were developed to account for the limited computational power available at the time, which would have made model-based approaches infeasible.

Model-based approaches are what we used in Chapter **??**, which involved fitting models the observed data and estimating effect sizes assuming that the data correspond to particular distributions. While computationally difficult for multivariate data, model-based approaches are potentially more powerful and less susceptible to error, at least when focussing on the effects of particular environmental drivers. An important question to ask when deciding between these general types of approaches is whether you are interested in observing/predicting patterns in data ('distance-based') or changes in variables ('model-based').

For responses exhibiting normal error distributions, it is appropriate to use MANOVA (multivariate analysis of variance). We will use the I x F data from Section **??** to demonstrate MANOVA. First, we need to convert the response variables in the dataframe into a matrix object.

```
hfeIxF <- read.csv("HFEIFplotmeans.csv")
str(hfeIxF)
```

```
## 'data.frame': 320 obs. of  5 variables:
##  $ plotnr  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date    : Factor w/ 20 levels "1/1/2008","1/1/2009",..: 3 13 7 5 15 2 6 17 1 19 ...
##  $ diameter: num  NA 3.96 7.38 NA 4.36 ...
##  $ height  : num  NA 4.05 6.18 NA 4.22 ...
##  $ treat   : Factor w/ 4 levels "C","F","I","IL": 4 4 4 4 4 4 4 4 4 4 ...
```

```
Y <- as.matrix(hfeIxF[,3:4])
m1 <- manova(Y ~ treat*Date, data=hfeIxF)
summary(m1)
```

```
##             Df  Pillai approx F num Df den Df    Pr(>F)
## treat        3 0.87681   53.083      6    408 < 2.2e-16 ***
## Date        16 1.83960  146.227     32    408 < 2.2e-16 ***
## treat:Date  48 0.97563    4.048     96    408 < 2.2e-16 ***
## Residuals  204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.aov(m1)
```

```
##  Response diameter :
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## treat         3  168.9  56.314  88.9642 < 2.2e-16 ***
## Date         16 4727.4 295.465 466.7754 < 2.2e-16 ***
## treat:Date   48  115.9   2.414   3.8136 1.417e-11 ***
## Residuals   204  129.1   0.633
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response height :
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## treat         3  150.1  50.042 134.6903 < 2.2e-16 ***
## Date         16 3461.7 216.358 582.3402 < 2.2e-16 ***
```

```
## treat:Date   48  157.1   3.273   8.8098 < 2.2e-16 ***
## Residuals   204   75.8   0.372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 48 observations deleted due to missingness
```

The analysis provides two sets of significance tests. Using `summary` gives test statistics and their significance for the multivariate response to the predictor variables. Using `summary.aov` gives univariate ANOVA tables for each response variable. We can see that both variables are responding to the treatments, which makes sense given the variables are strongly correlated.

Quite often, our data are not normally distributed (e.g., abundance data) and we rely on GLMs to fit models to appropriate error families. Recent advances in model-based approaches for multivariate data are included in the `mvabund` package. Let's reanalyse the Tibetan Plateau plant community data.

```
library(mvabund)
tibplat <- read.csv("tibplat.csv")
str(tibplat)
```

```
## 'data.frame': 32 obs. of  10 variables:
##  $ fertilization          : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ enclosure              : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ elymus_nutans          : num  3.53 4.85 2.2 2.87 3.53 ...
##  $ poa_crymophila         : num  0.89 0.6 0 0.3 1.79 ...
##  $ kobresia_setchwanensis : num  15.1 14.3 12.9 10.8 17.9 ...
##  $ anemone_rivularis      : num  5.12 13.65 0 18.76 5.12 ...
##  $ potentilla_fragarioides: num  3.62 3.62 2.36 3.15 2.05 2.52 2.05 1.57 0.86 0.22 ...
##  $ stipa_aliena           : num  3.8 3.8 0 1.52 6.08 5.32 4.56 7.6 0 0 ...
##  $ astragalus_polycladus  : num  2.08 2.08 2.08 0 1.04 8.3 0 0 0 0 ...
##  $ anemone_obtusiloba     : num  3.8 3.8 2.66 2.66 2.85 2.47 3.23 0.38 1.89 0 ...
```

```
X <- tibplat[,1:2]  # create matrix of predictor variables
Y <- mvabund(tibplat[,3:10])  # convert to 'mvabund' object, for use in downstream functions
plot(Y, ylab="")
```

```
## Kicking off BoxPlot sequence
```

```
## Overlapping points were shifted along the y-axis to make them visible.
```

```
##
##
## ABOUT TO PLOT THE FUNCTION
```

We can see in Fig. 1.25 that the data roughly fits a log-normal distribution based on the similar spread of the abundances for most species when plotted on a log scale. However, the data also contain several zero values. We will use a negative binomial error distribution, which will result in a warning since the variables represent biomass and not true abundances. A gamma distribution may be more appropriate here but is not available in the current release. Model diagnostics, as in Figure 1.26 can be viewed using the `plot` function on the model output.

```
# fit model
m1 <- manyglm(Y ~ fertilization*enclosure, data=X, family='negative.binomial')
```

```
## Warning in manyglm(Y ~ fertilization * enclosure, data = X, family = "negative.binomial"):
Non-integer data are fitted to the negative.binomial model.
```

```
# view model diagnostic plot
plot(m1)
```

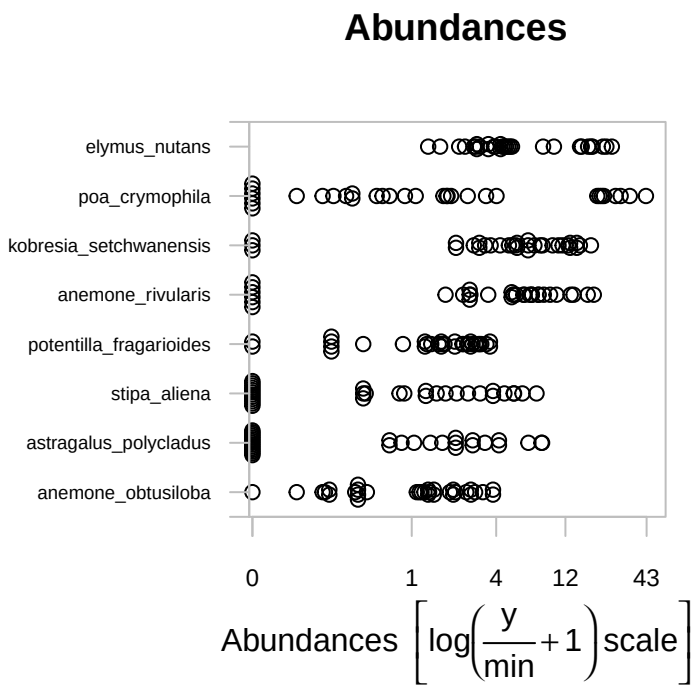

Figure 1.25: Species biomasses per plot in the Tibetan Plateau.

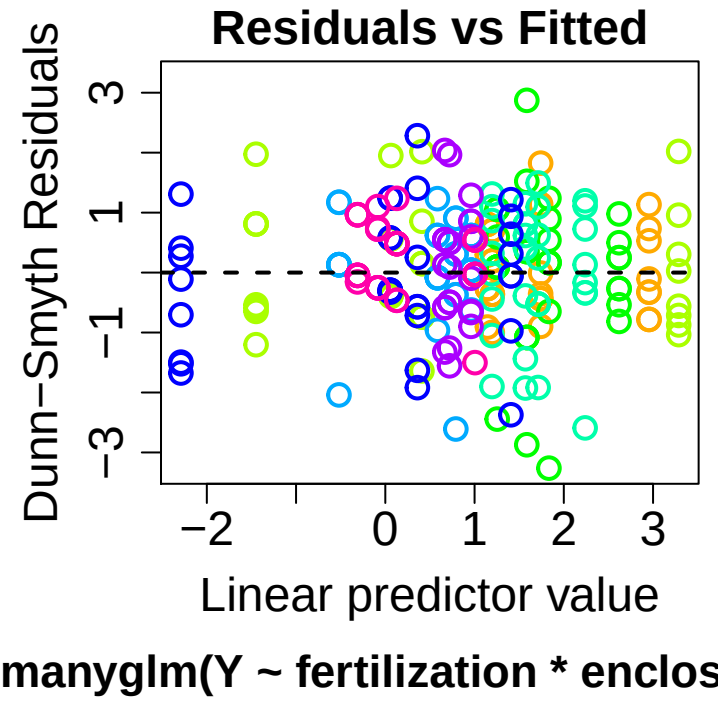

Figure 1.26: Model diagnostic plot for multivariate GLM fit to Tibetian plateau plant data.

```r
# look at model summary and ANOVA table
summary(m1)
```

```
##
## Test statistics:
##                          wald value Pr(>wald)
## (Intercept)                  13.753  0.000999 ***
## fertilization                 4.076  0.022977 *
## enclosure                     6.434  0.000999 ***
## fertilization:enclosure       5.372  0.000999 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Test statistic:   20.2, p-value: 0.000999
## Arguments:
##  Test statistics calculated assuming response assumed to be uncorrelated
##  P-value calculated using 1000 resampling iterations via pit.trap resampling (to account for correlat
```

```r
anova(m1, nBoot=100, p.uni='adjusted')  # we use only 100 permutations here, to save time
```

```
## Time elapsed: 0 hr 0 min 0 sec
## Analysis of Deviance Table
##
## Model: manyglm(formula = Y ~ fertilization * enclosure, family = "negative.binomial",
## Model:     data = X)
##
## Multivariate test:
##                          Res.Df Df.diff   Dev Pr(>Dev)
## (Intercept)                  31
## fertilization                30       1 75.03     0.01 **
## enclosure                    29       1 80.96     0.01 **
## fertilization:enclosure      28       1 55.17     0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Univariate Tests:
##                          elymus_nutans          poa_crymophila
##                                    Dev Pr(>Dev)            Dev Pr(>Dev)
## (Intercept)
## fertilization                   29.644    0.010          14.06    0.010
## enclosure                       23.598    0.010         37.826    0.010
## fertilization:enclosure         12.118    0.020          1.498    0.545
##                          kobresia_setchwanensis          anemone_rivularis
##                                             Dev Pr(>Dev)               Dev
## (Intercept)
## fertilization                            13.008    0.010             2.129
## enclosure                                11.308    0.010             3.473
## fertilization:enclosure                    2.37    0.426             0.061
##                                  potentilla_fragarioides
##                          Pr(>Dev)                     Dev Pr(>Dev)
## (Intercept)
## fertilization               0.198                    3.81    0.099
## enclosure                   0.346                   0.557    0.743
## fertilization:enclosure     0.743                  11.274    0.020
##                          stipa_aliena          astragalus_polycladus
```

```
##                                   Dev Pr(>Dev)                    Dev
## (Intercept)
## fertilization                   4.823    0.089                  0.485
## enclosure                       0.016    0.921                  2.688
## fertilization:enclosure        15.039    0.010                  8.987
##                                   anemone_obtusiloba
##                          Pr(>Dev)                Dev Pr(>Dev)
## (Intercept)
## fertilization               0.436              7.075    0.050
## enclosure                   0.386              1.496    0.505
## fertilization:enclosure     0.040               3.82    0.228
## Arguments:
##  Test statistics calculated assuming uncorrelated response (for faster computation)
## P-value calculated using 100 resampling iterations via PIT-trap resampling (to account for correlatio

# (not all output shown)
```

The first part of the `anova()` result gives the significance of the predictor variables, while the second part gives their significance in univariate tests, to determine which species are particularly responsive.

Large datasets with lots of species can lead to calls to `summary()` and `anova()` taking a very long time. An alternative approach to assessing the significance of treatments and/or their interactions is via sequential model selection and assessment using the `drop1` function. The output includes AIC scores for the global model ("<none>") and each model for which a single term can be dropped. If the AIC is lower for any of the latter models than it is for the global model, this is evidence in support of the simpler model and justifies attempting to simplify the model further. Let's demonstrate using all plant species in the Tibetan Plateau dataset.

```
# set up predictor and response tables
X <- tibplat[,1:2]  # create matrix of predictor variables
Y <- mvabund(tibplat[,3:ncol(tibplat)])  # convert to 'mvabund' object, for use in downstream functions

# fit model (ignore warning)
m2 <- manyglm(Y ~ fertilization*enclosure, data=X, family='negative.binomial')

## Warning in manyglm(Y ~ fertilization * enclosure, data = X, family = "negative.binomial"):
Non-integer data are fitted to the negative.binomial model.

# evaluate effect of dropping interaction from model on overall fit (ignore warning)
drop1(m2)

## Warning in manyglm(formula = Y ~ fertilization + enclosure, family = "negative.binomial",
:  Non-integer data are fitted to the negative.binomial model.

## Single term deletions
##
## Model:
## Y ~ fertilization * enclosure
##                          Df    AIC
## <none>                      1008.1
## fertilization:enclosure  8 1048.7
```

The AIC score is much higher when the `fertilization:enclosure` term is removed. Therefore, we conclude that the global model containing both fixed effects and the interaction performs best.

1.5 Functions used in this chapter

For functions not listed here, please refer to the index at the end of this book.

All functions in this table are from the `vegan` package unless otherwise noted.

Function	What it does	Example use
`summary` (base R)	Simple summary table for dataframes and `rda` and `cca` objects	`summary(pupae)`
`rda`	Principal components analysis (as used in this chapter), and redundancy analysis	Section 1.3
`cca`	Correspondence analysis. Similar to `rda`, but using χ^2 distances to estimate divergence between samples (and thus appropriate for count data)	Section 1.3
`decorana`	Estimate the 'gradient length', to decide between PCA or CA	Section 1.3.2.1
`wcmdscale`	Principle coordinates analysis, used in conjunction with `vegdist`	Section 1.3.3
`vegdist`	Calculate a distance matrix, used primarily as input to `wcmdscale`, `mantel` and `hclust`	Section 1.3.3
`rankindex`	Compare dissimilarity indices for gradient detection	Section 1.3.3
`metaMDS`	Non-metric multi-dimensional scaling. Suitable for a wide variety of data.	Section 1.3.4
`scores`	Extract species or site loadings from a fitted NMDS or other ordination method.	Section 1.3.4
`mantel`	Test for correlation between two dissimilarity matrices	Section 1.4.1
`envfit`	Identify variables from an environmental matrix that are correlated with the site loadings	Section 1.4.2.1
`ordistep`	Use forward and/or backward selection to select environmental variables associated with variation in the response table	Section 1.4.2.1
`varpart`	Partition variation in a response table to variables in two or more tables of explanatory variables	Section 1.4.3
`pcnm`	Calculate principal coordinates of neighbour matrices for partitioning variation to spatial patterns	Section 1.4.3.1
`hclust`	Hierarchical clustering on a dissimilarity matrix	Section 1.4.5.1
`adonis`	Permutational multivariate analysis of variance with distance matrices (PerMANOVA)	Section 1.4.5.1
`manova`	(base R) Multivariate analysis of variance	Section 1.4.5.3
`manyglm`	(`mvabund` package) Generalized linear models for multivariate abundance data	Section 1.4.5.3
`mvabund`	(`mvabund` package) Construct a multivariate abundance object, for use in e.g. `manyglm`	Section 1.4.5.3

1.6 Exercises

In these exercises, we use the following colour codes:

- ■ **Easy**: make sure you complete some of these before moving on. These exercises will follow examples in the text very closely.
- ♦ **Intermediate**: a bit harder. You will often have to combine functions to solve the exercise in two steps.
- ▲ **Hard**: difficult exercises! These exercises will require multiple steps, and significant departure from examples in the text.

We suggest you complete these exercises in an **R** markdown file. This will allow you to combine code chunks, graphical output, and written answers in a single, easy-to-read file.

1.6.1 Alpha diversity

1.6.1.1 Soil fungal data

1. ■ Using the data from 'IxFsub.csv', produce boxplots showing observed and rarefied richness as a function of `Harvest` and `Treatment`.
2. ■ Fit a linear model and use ANOVA to test for effects and interactions of `Harvest` and `Treatment` on observed and rarefied richness.
3. ■ The experimental design is actually nested, with samples collected from around two trees in each plot on multiple timepoints. Use `lmer` in the `lme4` package to fit the models in the previous question to also include `Plot` and `Tree` as random effects. Inspect variation partitioned to random effects and evaluate whether the fixed factors are significant.

1.6.2 Ordination

1.6.2.1 Endophyte data

1. ■ Read in the data from 'endophytes.csv'; see Section **??** (p. **??**) for a description of the data. Use the `decorana` function to calculate gradient lengths and determine whether PCA is appropriate for these data (see Section 1.3.2.1 for help, if necessary).
2. ♦ Perform CA these data and plot the results. Notice the strong skew in the data along both axes. Try again after standardising the community matrix using the `decostand` function (try the 'hellinger' and 'max' methods). Notice the undesireable parabolic pattern in the ordination and strong skew; this suggests that CA is not an improvement over PCA (common for data matrices that contain many zeros, collected along long environmental gradients).
3. ♦ Perform PCoA on these data, using the 'hellinger' method for the `decostand` function and 'bray' method for the `vegdist()` function, and plot the results. See Section 1.3.3 for help, if necessary. Repeat as before but use the `binary` argument in the `vegdist` function to convert the matrix to a 'presence/absence' matrix.

1.6.3 Analysis of Structure 1: two-table analysis

1.6.3.1 Endophyte data

1. ♦ Look at the help page for the `capscale` function. Use `capscale` to perform distance-based RDA (constrained PCoA) using the continuous variables in 'endophytes_env.csv' (percentC, percentN, CNratio) as predictors, then plot the results. First use the `envfit` function to determine which variables to include in db-RDA (Section 1.4.2).
2. ♦ Repeat the analysis in the previous exercise but use the `ordistep` function to determine which variables to include in db-RDA.

1.6.4 Analysis of Structure 2: variation partitioning

1.6.4.1 Endophyte data

1. ♦ Perform variation partitioning to determine whether leaf species, leaf chemistry, or sample type explains the most variation in fungal community composition.
2. ♦ Use the geographic coordinates of each plot to estimate the contribution of space to variation in fungal community composition. Is this estimate greater than the variation partitioned to the measured leaf variables?
3. ♦ Test the significance of each individual partition.
4. ♦ Generate dummy variables (using 'dudi.hillsmith') for each of the levels of 'species' and check whether there are particular leaf species that explain variation in fungal community composition.

1.6.5 Analysis of Structure 3: 'experimental' systems

1.6.5.1 Allometry data

1. ♦ For the allometry data, plot a dendrogram of multivariate distances (euclidean) among individual trees based on the four growth parameters, labelling the tips of the dendrogram with the species level. Use ANOSIM and PERMANOVA to test the hypothesis that clusters can be explained by interspecific variation. See Section 1.4.5.1 for help, if necessary.
2. ♦ Using your knowledge from Chapter **??** and Sections 1.3.4 and 1.4.5.1, plot the ordination results using coloured circles to represent the different tree species and include a legend.
3. ▲ Overlay the plot with the centroid (i.e., average) for each species, using a different symbol than for the individual points. Modify the axes to reflect the percentage of inertia (i.e., variance) explained by each axis. Refer to Chapter **??** for help tabulating mean values, if necessary.

1.6.5.2 Endophyte data

1. ♦ Use the `adonis` function to test for main and interactive effects of tree species and tissue type (fresh vs litter) on fungal community composition (see Section 1.4.5.1). The predictor variables can be found in 'endophytes_env.csv'. What terms were significant? Which term explained the most variation?

2. ▲ Plot the PCoA results and use different symbols and colours to reflect the identities of tree species and tissue types. Add a legend to the plot. Use information from Chapter **??** and Sections 1.3.4 and 1.4.5.1 for help, if necessary. Hint: automatic functions for generating vectors of colours, such as `rainbow`, can lead to very similar colours with so many treatment levels. Check out the `randomcoloR` package for alternative approaches.
3. ▲ Overlay the plot with ellipses representing 95% confidence intervals for each species and sample type using functions seen in section 1.4.5.1.

1.6.5.3 Mite data

1. ■ Use `manova` to estimate the responses of mite community composition to the environmental variables associated with the `mite` data.
2. ⧫ There is not a built in function to view diagnostic plots for `manova` output. Use the `resid` function to obtain the residuals for each response variable and use the `qqnorm` and `qqline` functions to produce quantile-quantile plots for a few of the response variables to determine whether it is appropriate to model the responses using a normal error distribution.
3. ⧫ Use `manyglm` to estimate the responses of mite community composition to the environmental variables associated with the `mite` data. Which error family, poisson or negative binomial, provides the best fit to the data? Look at the results of the best fitting model.

Index