Practical 5: Unconstrained Ordination

BIO2020 Statistics and experimental design

Introduction

Until now, all the 'response' data you have looked at consists of a single column of data. Examples you have looked at include 'growth', 'presence or absence of bacterial colonies' etc. You can easily summarise data that consists of a single column of numbers using a mean, median, standard deviation. But what if your response variable consists of lots of columns of data? For example:

- instead of a single column telling you the total number of specis found at 20 sites, you have a table of 20 rows, with 17 columns. Each cell entry in the table shows the abundance of a particular species (column) at that site (row)
- instead of single column giving an indication of the number of bacteria colonies you have multiple columns, from qPCR sequencing, giving information about the genetic information (each column) for each bacterial colony (each row)

Such tables of data are harder to understand using means and standard devations. Whilst you can still calculate these, you end up with lots of information to digest. For example, you can still calculate the average abundance of each of your 17 species at all your 20 sites. But this does not allow you to answer key questions, e.g.

- which sites are most similar in their species composition?
- which species regularly co-occur with each other?
- which bacterial strains have the most similar genomes based on the qPCR results?

Analyses of tables of data with multiple columns, rather than a single column, are sometimes called "multivariate analyses", to distinguish them from the "univariate analyses" you covered in Practicals 1 to 4. These tables can be formally analysed with the aid of explanatory data, which we will cover in Practical 6. In this practical we will focus on methods to simplify the data, so that they can be easily visualised, and relationships between rows and columns of your input data better understood. These methods are known as **unconstrained ordination**. Here we say "unconstrained" because we do not directly incorporate explanatory variables into the analysis, although after the initial analysis has been done, we can use them to aid interpretation. We say "ordination" because we can create metrics to order the rows and columns of your input data on the basis of their similarity.

Types of ordination

Ordination is a way of arranging rows in your table (e.g. sites or samples) along gradients in such a way that we can try and explain patterns of variation within the noise. Hopefully we can explain these gradients on the basis of some explanatory data **after** we have done the ordination (hence **unconstrained** ordination). Linear methods (**PCA**) assume the observations respond roughly linearly along these gradients. Unimodal methods (**CA**) assume the observations will respond non-linearly. Distance-based models (**NMDS**) use the relative rankings of your sites along each axis.

Unconstrained ordination is a useful tool for determining whether there is a relationship between multiple **response variables**. Through measuring the similarity in their **composition**, it is easier to compare samples. When the composition of samples is summarised on a continuous scale it is referred to as an **ordination**, whereas on a categorical scale it is known as **classification** or **clustering**.

The main aims of this practical are to learn more on the use, interpretation and graphical visualisation of unconstrained ordination. Specific objectives are to learn how to use:

1. Principal components analysis (PCA), a "linear" method
2. Correspondence analysis (CA), a "unimodal" method
3. Non-metric multidimensional scaling (NMDS), a "rank-based" method

You are strongly recommended to study this interactive website before beginning the practical. We will use the same data as on the website, but we have swapped them round in the analyses, to keep you thinking!

Setting up for the practical

First, go to Canvas and download the **dune.csv** data set and save it to your `Data` folder within your BIO2020 project folder. Now navigate to your BIO2020 folder and open up your existing project for this course in RStudio, as you have done in previous practicals. On the main RStudio menu, click on **File -> New File -> R script** and save it to your current working directory (the location of your Bio2020 project) as **Practical_5.R**

Installing and Loading Packages

Practicals 5 and 6 make use of both the `vegan` and `bio2020` R packages. Hopefully you have already installed these. The `vegan` package was originally written for vegetation analysis, but is now used in microbiology, phylogenetics, animal ecology etc. There is plenty of online material about the package, but if using Google to search for it, please search for 'R vegan multivariate' rather than just 'vegan' otherwise Google will return food recipes!

If you have not already done so, please install the `bio2020` package; this provides easier access to analytical functions, and better graphics, than the standard `vegan` functions. Remember that you have to set your R session to look in the correct folder (e.g. Downloads) to install the packages. There is extensive information on how to install the package, including troubleshooting for Safari on MacBooks, at Canvas on this page . The `bio2020` package automatically activates the `vegan` package.

Load the **bio2020** package as you have done in previous sessions.

```
library(bio2020)
```

Example datasets for analysis

Import the `dune.csv` file you downloaded at the very beginning of the practical into R using the `read.csv()` function. This practical assumes you have save your file in the `Data` subfolder:

```
dune <- read.csv("Data/dune.csv", row.names=1)
```

Vegan has some in-built data files, let's load "varespec" and "varechem".

```
data(varespec)
data(varechem)
```

These datasets are plant ecological ones, that are often used in online tutorials about the `vegan` package. Here each column is a species, and each row a sample. Your dataset would have the identical structure if dealing with e.g. genomic data, each column might indicate a sequence mutation insert or deletion, and each row a sample. For an example of how the `vegan` package has been extended to handle complex genetic data, including **operational taxonomic units** (OTUs), output from **mothur** and **QIIME** look at the excellent phyloseq R package.

Summary Statistics

It is always good to start by observing the data and running some summary statistics.

```
summary(varespec)

summary(dune)
```

However, you will notice that this output is rather large, hence why it is not included here. With big datasets it can be easier to observe the data by simply using **head()** to get a snapshot of the data, and then **nrow()**, **ncol()**, **rownames()**, and **colnames()** to explore the data a bit more.

You can also use the `View()` function (capital 'V') within RStudio to show it as a grid.

```
# Show first few lines and check dimensions
head(varespec)
nrow(varespec)
ncol(varespec)

# Column names are abbreviated Latin names
rownames(varespec)
colnames(varespec)

# Similarly for dune dataset
head(dune)
nrow(dune)
ncol(dune)
rownames(dune)
colnames(dune)

# Display in spreadsheet-like tab
View(dune) # not view(dune)
View(varespec)
```

1. Principal Components Analysis

1.1 PCA analysis and summary of results

For the purpose of clarity the wrapper function **ordi_pca()** in the package **bio2020** will be used in this practical. A typical PCA in R has this format: **ordi_pca(data)**

In the online tutorial you saw how the arch or horseshoe effect can be a problem with ecological or gene sequencing data. Although **unimodal methods** are a good alternative, you can implement transformations using the **decostand()** function. Here, we use **scale = TRUE** to give the data **unit variance**, and the **hellinger** transformation, which works well with environmental and species data.

```
varespec_pca <- ordi_pca(decostand(varespec, method = "hellinger"), scale = TRUE)
varespec_pca
```

```
## Call: rda(X = spp_data, scale = TRUE)
##
##               Inertia Rank
## Total              44
## Unconstrained      44   23
## Inertia is correlations
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
## 8.603 5.134 4.576 3.714 3.245 2.779 2.626 2.221
```

```
## (Showing 8 of 23 unconstrained eigenvalues)
```

Output explained:

- **Call** - This is a reminder of how you fitted the PCA.
- **Inertia** - Total inertia is a measure of total variance.
- **Unconstrained** - How many correlations have been explained by the ordination axes. In this case, there is one less than the number of variables.
- **Eigenvalues** - Measure of the amount of variance that is explained by each of the axes
- **PC1** - The first axes, and will explain the largest amount of variation of all the axes.

To see all 23 axes, use **summary()**. This is a large output so it is not included here.

```
summary(varespec_pca)
```

This produces a lot of output, but don't worry, it is less fearsome than it looks!

Output explained:

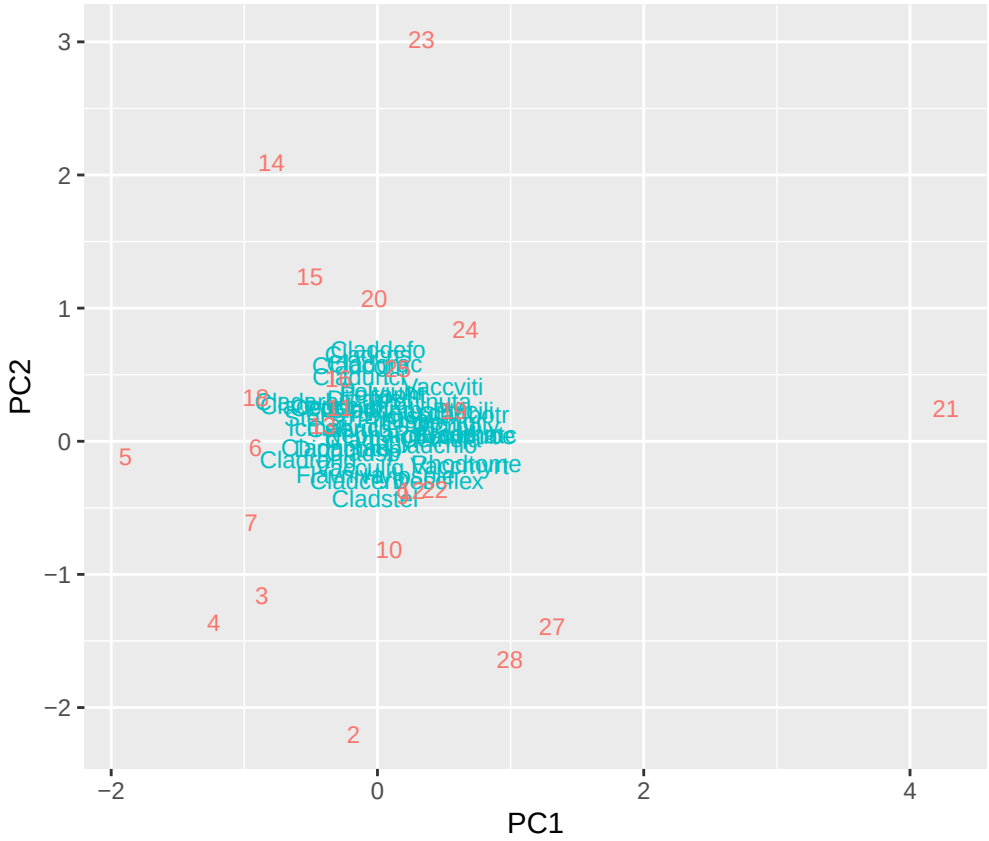
- The top 6 lines are the same as the PCA output
- **Eigenvalues** are the amount of inertia (variance) explained by each axis. If added up for all 23 axes, it will be the same as the total inertia.
- **Proportion explained** is the amount of total variance explained, and if added up for all 23 axes will equal 1.00. You can see that the proportion explained is highest for PC1, next highest for PC2 and so on. The last PC, PC23, explains virtually nothing.
- **Cumulative proportion** is the cumulative total of the proportion of each axes. While **PC1** explains 0.5384 and **PC2** explains 0.2543 of the variance, their *cumulative* proportion is 0.7927 because it is the total amount of variance they explain together.
- **Species and Site scores** - the relative weight each species and/or site has within each axes.

The lines I find most useful to focus on are the Proportion explained, and Cumulative proportion. PCA tries to "squash" all the original data from 24 dimensions (we have 24 columns or species) into 2 dimensions. Why? You cannot plot a 24 dimensional graph (or if you know how to do so, please apply for a Nobel Prize!). But you can plot a 2 dimensional graph. So if you can squash as much of the 'information' in your original table of data into 2 new variables that summarise your species that makes life easier.

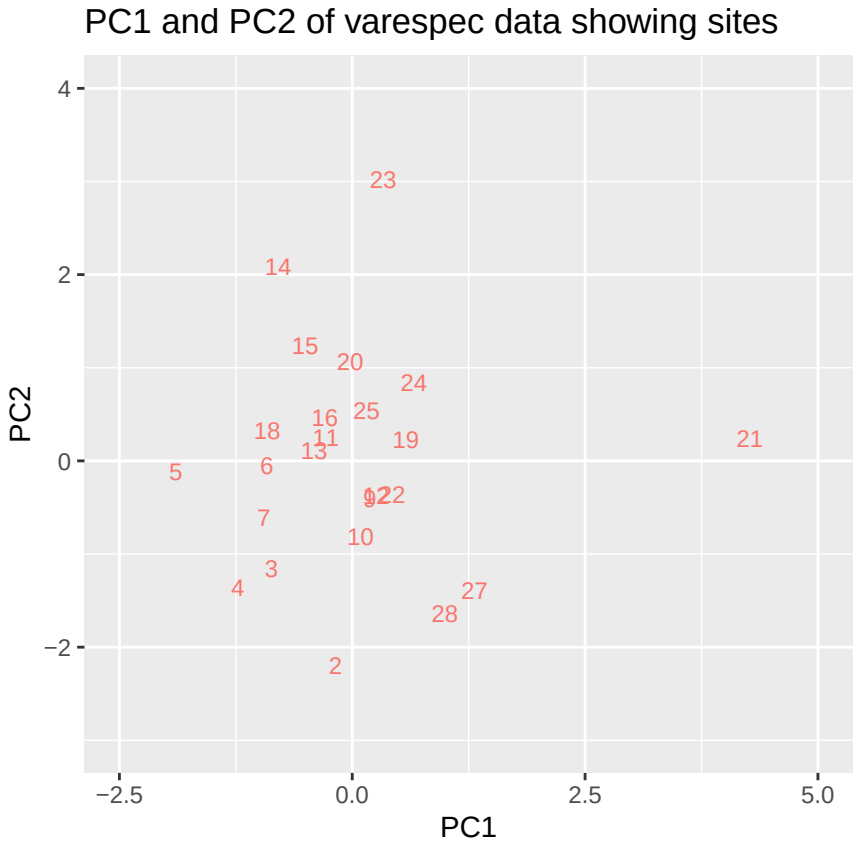
1.2 Visualise your PCA results

In general we are only interested in PC1 and PC2. Very rarely, there might be useful information in PC3 so you may want to check it, although in my experience it is usually of little value. Generally, the first two axes tend to explain the majority of the variation, so these are usually plotted. We will use the `ordi_plot()` function from the `bio2020` package:

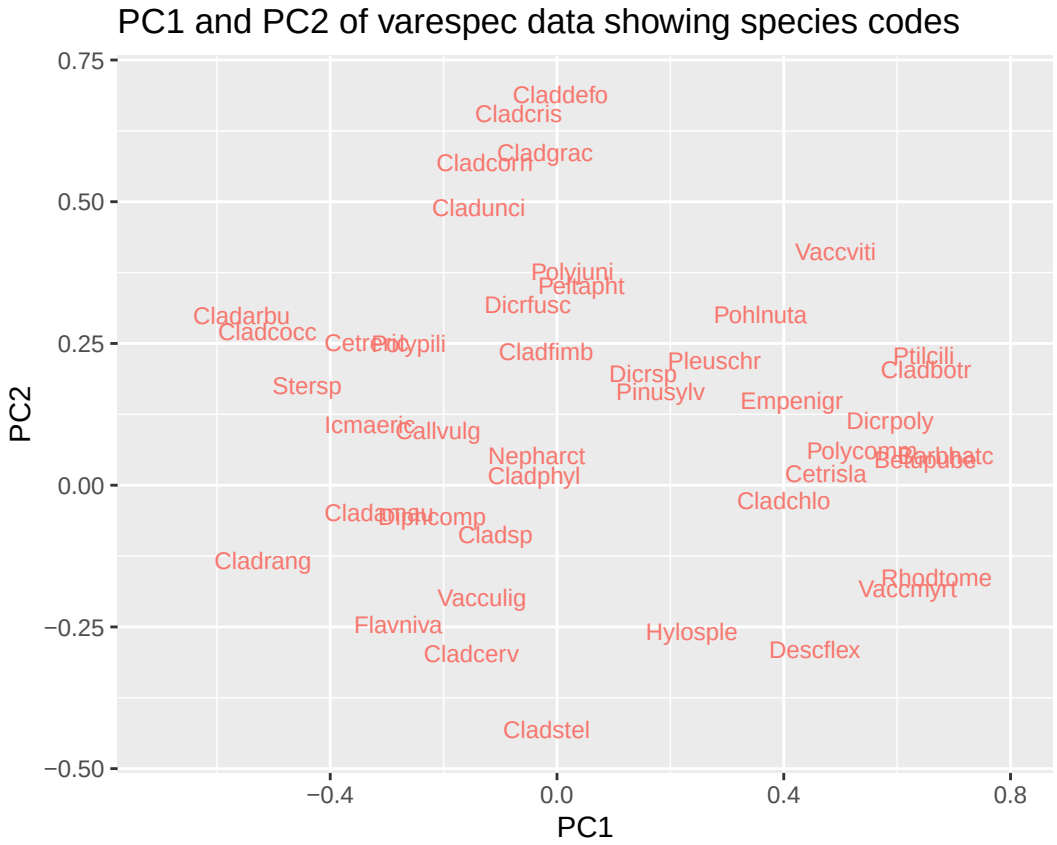
```
# both sites and species
ordi_plot(varespec_pca, geom = "text")
```




```
# just sites; note how you control x and y axis limits
ordi_plot(varespec_pca, layers="sites", geom="text") %>%
  gf_labs(title="PC1 and PC2 of varespec data showing sites") %>%
  gf_lims(x = c(-2.5, 5), y=c(-3, 4)) # optional
```




```
# just species
ordi_plot(varespec_pca, layers="species", geom="text") %>%
  gf_labs(title="PC1 and PC2 of varespec data showing species codes") %>%
  gf_lims(x = c(-0.7, 0.8), y=c(-0.45, 0.7))
```



When this dataset was analysed initial on the interactive website with PCA it suffered from a severe 'arch effect'. You will notice we do not have an arch effect now, even though we are still using PCA. This is because we used `decostand()` to standardise our data, and reduce the effects of outliers. Overall PC1 and PC2 explain less variation than before but the arch effect has been removed so the graph is easier to understand.

Species that are close together tend to co-occur at particular sites, and sites that are close together have similar species composition. Species that are close to the zero-zero lines are more common across all sites. You will notice that due to clustering it is difficult to make some species/sites, so you can use **ordi_identify()** to pick out the interesting ones.

After you enter the `ordi_identify()` command you will see the text

```
"Click on plot to label points; hit Esc key to exit"
```

displayed in the Console:

```
# Create the species plot, but only use points, not text labels
# store the species plot in varespec_plt
varespec_plt <- ordi_plot(varespec_pca, layers="species", geom="point")

# Displays the species plot onscreen
varespec_plt

# Use mouse to identify individual species. Hit Esc to exit
ordi_identify(varespec_plt)
```

The `ordi_identify()` function is still under development so may not detect every point. In the Console, type `?ordi_identify` to see more examples

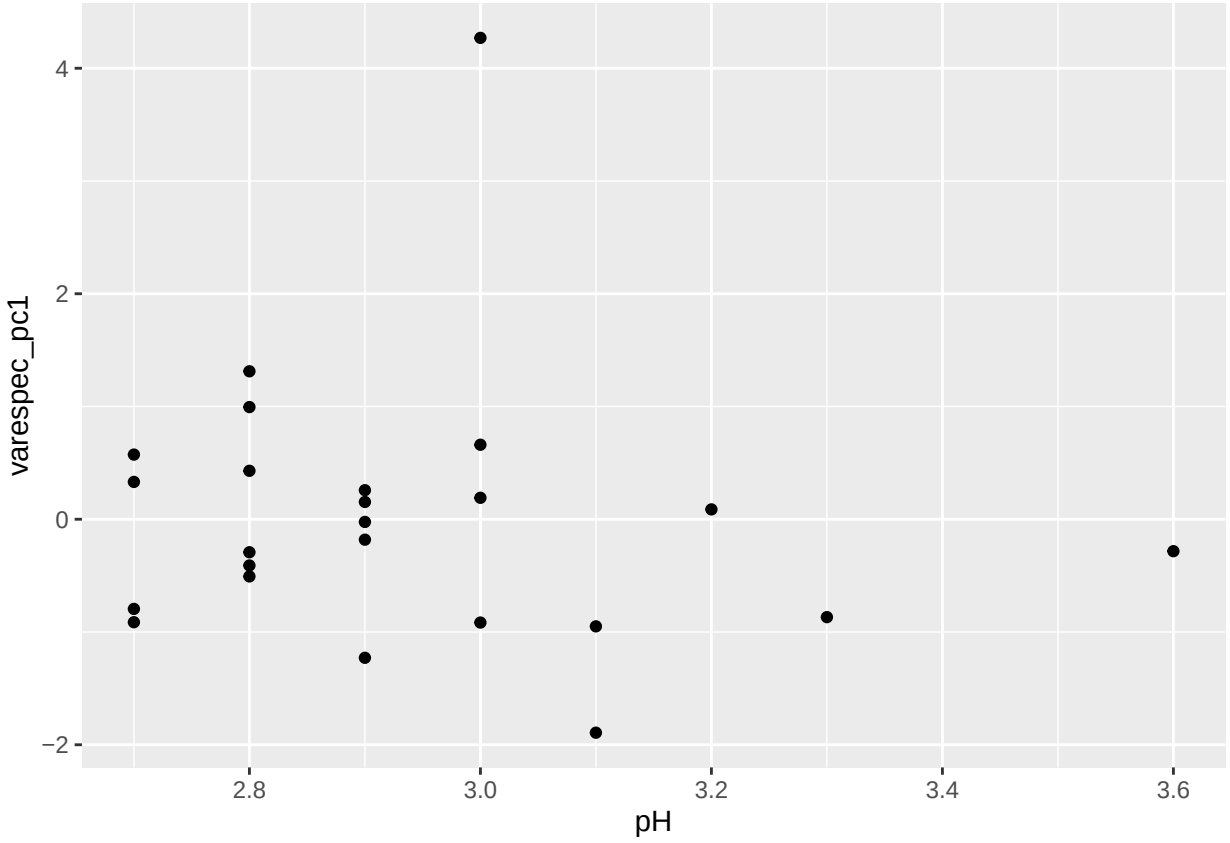
1.3 Relate PCA to explanatory variables

In unconstrained methods you undertake the ordination analysis first, and then relate your results back to any explanatory variables later. First, you need to obtain the sample PC scores used in your earlier plot. These are `choices = 1` for PC1 (x-axis in PC1 v PC2 plot) or `choices = 2` for PC2.

After having obtained these scores, you can see if there is any relationship with the explanatory data (in this case soil chemistry), to help you understand what external factors might be affecting the results. Your explanatory data in genetic experiments might be different stressors, resistant or susceptable strains etc.

```
# Extract the PC1 site scores
varespec_pc1 <- scores(varespec_pca, display="sites", choices = 1)

# Create plot (point plot because pH is continuous data)
# We put soil Ph on the horizontal x-axis as we assume it determines the
# plant species composition
gf_point(varespec_pc1 ~ pH, data=varechem)
```



Think about how the PC1 scores relate to the environment, and try the same with PC2, or with some of the other soil chemistry variables.

2. Correspondence Analysis

2.1 CA analysis and summary of results

The wrapper function **ordi_ca()** in the package **bio2020** will be used in this practical. A typical PCA in R has this format: **ordi_ca(data)**

Correspondence analysis is a weighted form of PCA that can fit non-linear responses. The weighting means the analysis is on the relative composition instead of absolute values. It is similar to PCA in that species that are close together tend to co-occur, and sites that are close together tend to have similar species composition.

```
dune_ca <- ordi_ca(dune)
dune_ca
```

```
## Call: cca(X = spp_data)
##
##               Inertia Rank
## Total           2.115
## Unconstrained   2.115   19
## Inertia is scaled Chi-square
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
## 0.5360 0.4001 0.2598 0.1760 0.1448 0.1079 0.0925 0.0809
## (Showing 8 of 19 unconstrained eigenvalues)
```

Output explained (similar to PCA):

- **Call** - This is a reminder of how you fitted the CA.
- **Inertia** - Total inertia is a measure of total variance.
- **Unconstrained** - How many correlations have been explained by the ordination axes. In this case, there is one less than the number of variables.
- **Eigenvalues** - Measure of the amount of variance that is explained by each of the axes

To see the full output, use **summary()**. Again, this is a large output so it is not shown here.

```
summary(dune_ca)
```

Output explained (similar to PCA):

- **CA1** - The first axes, and will explain the largest amount of variation of all the axes.
- The top 6 lines are the same as the CA output
- **Eigenvalues** are the amount of inertia (variance) explained by each axis. If added up for all 23 axes, it will be the same as the total inertia.
- **Proportion explained** is the amount of total variance explained, and if added up for all 23 axes will equal 1.00.
- **Cumulative proportion** is the cumulative total of the proportion of each axes. While **CA1** explains 0.2534 and **CA2** explains 0.1892 of the variance, their *cumulative* proportion is 0.4426 because it is the total amount of variance they explain together.
- **Species and Site scores** - the relative weight each species and/or site has within each axes.

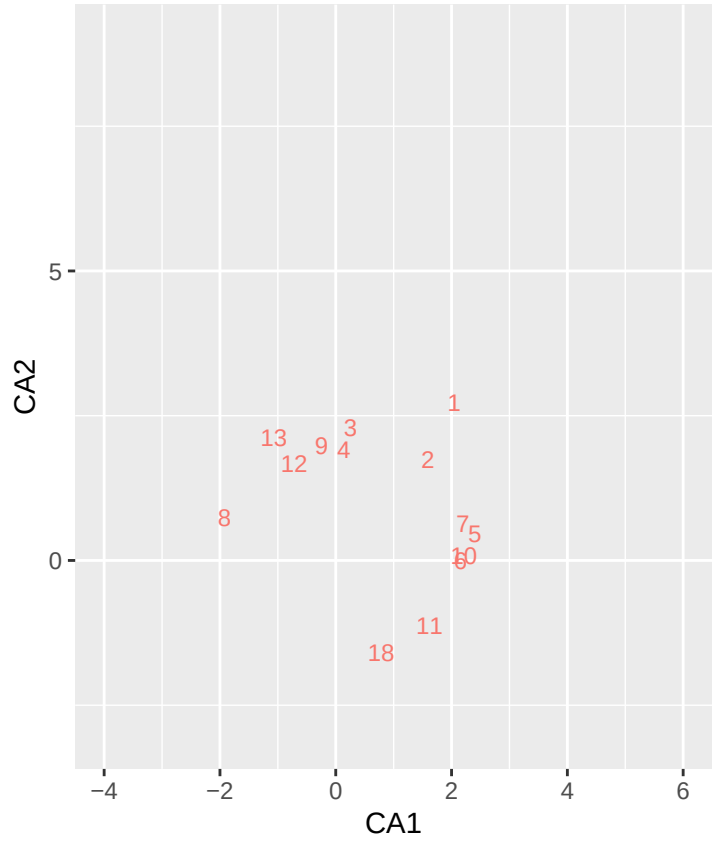
You can see that whilst PCA and CA differ in their underlying philosophy (linear vs unimodel) the summary of the results of the analyses is similar in interpretation.

2.2 Visualise your CA results

Visualisation is very similar to that for PCA, and is most easily accessed via `ordi_plot()` which is compatible with the plotting system you have already used.

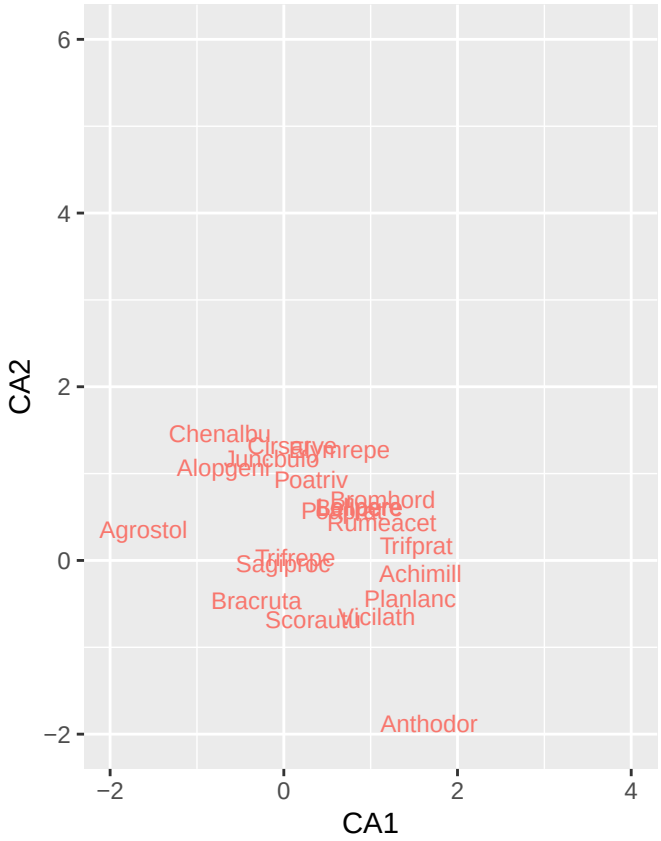
```
# All the rows (sites or samples)
ordi_plot(dune_ca, layers = "sites", geom = "text") %>%
  gf_lims(x = c(-4, 6), y=c(-3, 9))
```

```
## Warning: Removed 6 rows containing missing values (`geom_text()`).
```




```
# All the columns (species or attributes)
ordi_plot(dune_ca, layers = "species", geom = "text") %>%
  gf_lims(x = c(-2, 4), y=c(-2, 6))
```

```
## Warning: Removed 9 rows containing missing values (`geom_text()`).
```

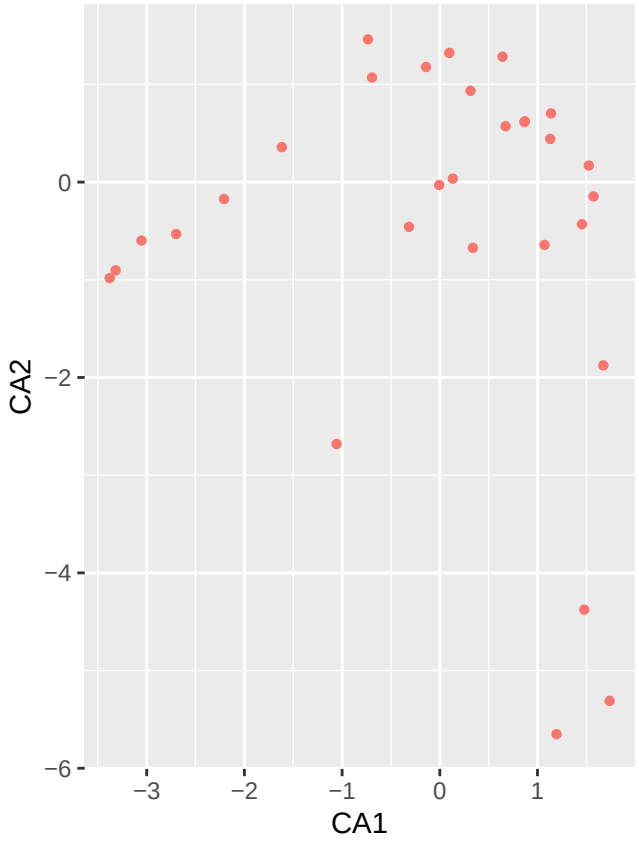


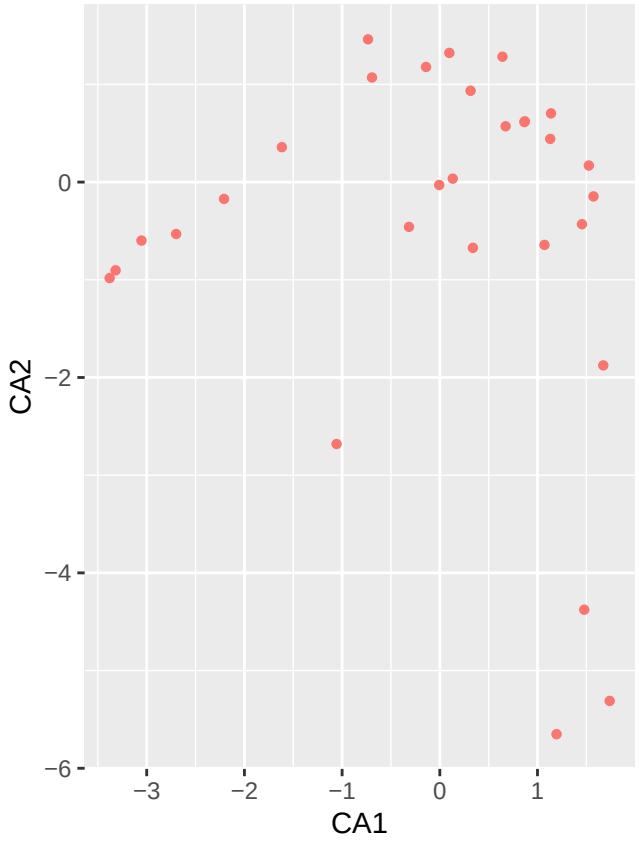
Again, the species or attribute plot can get very cluttered depending on your dataset, so you might find it easier to label individual points using `ordi_identify()`:

```
# Create the species plot, but only use points, not text labels
# store the species plot in dune_plt
dune_plt <- ordi_plot(dune_ca, layers="species", geom="point")

# Displays the species plot onscreen
dune_plt

# Use mouse to identify individual species. Hit Esc to exit
ordi_identify(dune_plt)
```


0
−2
−4
−6
CA2
−3
−2
−1
0
1
CA1



If you have time, try plotting axes 1 and 3, CA1 vs CA3, using `axes = c(1,3)` as an option to `ordi_plot()`.

3. Non-metric Multidimensional Scaling

3.1 NMDS background and analysis

NMDS creates the ordination using the rank order of your sites along each axis, rather than absolute multi-dimensional distances. NMDS only makes sure that the points further apart are still further apart in NMDS space than the "closer together" points, so it does not preserve the actual distance. It determines these rankings from pairwise similarity scores between each pair of samples in turn, and analysing the resultant table. The **ordi_nmds()** function by default uses the Bray-Curtis similarity measure which is robust for most data. The algorithm has to run multiple times to find the best solution. If needed, it will automatically standardise your data by either square-root and/or 'Wisconsin' standardisation, which standardises species to equal maxima and sites to equal totals. You may see this displayed in the output as the model runs.

The wrapper function **ordi_nmds()** in the package **bio2020** will be used in this practical. A typical PCA in R has this format: **ordi_nmds(data)**

```
dune_nmds <- ordi_nmds(dune)
```

```
## Run 0 stress 0.1192678
## Run 1 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 0.02027111  max resid 0.0649656
## Run 2 stress 0.1939202
## Run 3 stress 0.2496649
## Run 4 stress 0.1183186
```

```
## ... New best solution
## ... Procrustes: rmse 1.306338e-05  max resid 4.160223e-05
## ... Similar to previous best
## Run 5 stress 0.1192678
## Run 6 stress 0.2253072
## Run 7 stress 0.1192678
## Run 8 stress 0.1192678
## Run 9 stress 0.1192678
## Run 10 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 2.952944e-06  max resid 9.465589e-06
## ... Similar to previous best
## Run 11 stress 0.1886532
## Run 12 stress 0.1183186
## ... Procrustes: rmse 2.138015e-06  max resid 4.929026e-06
## ... Similar to previous best
## Run 13 stress 0.1183186
## ... Procrustes: rmse 4.554984e-06  max resid 1.54301e-05
## ... Similar to previous best
## Run 14 stress 0.1183186
## ... Procrustes: rmse 6.698982e-06  max resid 1.941852e-05
## ... Similar to previous best
## Run 15 stress 0.1812933
## Run 16 stress 0.1192679
## Run 17 stress 0.1192679
## Run 18 stress 0.1809577
## Run 19 stress 0.1183186
## ... Procrustes: rmse 2.756427e-06  max resid 9.340307e-06
## ... Similar to previous best
## Run 20 stress 0.1808911
## *** Best solution repeated 5 times
```

Output explained:

NMDS uses many random starts (20 each time) and looks for the fits with the lowest stress. It will only conclude that a solution has been reached when the solutions with the lowest stress are similar. It also fits the NMDS for 1,2,3...etc dimensions, and stops after a sudden drop in stress is observed. Sometimes the NMDS cannot find a solution on the first try like it has here (you will get back **no convergent solutions**), which is why we've saved the output as the object **dune_nmds**. To run the NMDS on the same data again:

dune_nmds_2 <- ordi_nmds(dune, previous.best = dune_nmds)

The **previous.best** argument passes in a previous fit of NMDS and will run another 20 random starts (making the total number of random starts 40). However, you do not need to run this on the dune dataset as we have already found a convergent solution.

Check the output of the nmds:

```
# Check the output
print(dune_nmds)
```

```
##
## Call:
## metaMDS(comm = spp_data)
##
## global Multidimensional Scaling using monoMDS
##
```

```
## Data:     spp_data
## Distance: bray
##
## Dimensions: 2
## Stress:     0.1183186
## Stress type 1, weak ties
## Best solution was repeated 5 times in 20 tries
## The best solution was from try 10 (random start)
## Scaling: centring, PC rotation, halfchange scaling
## Species: expanded scores based on 'spp_data'
```

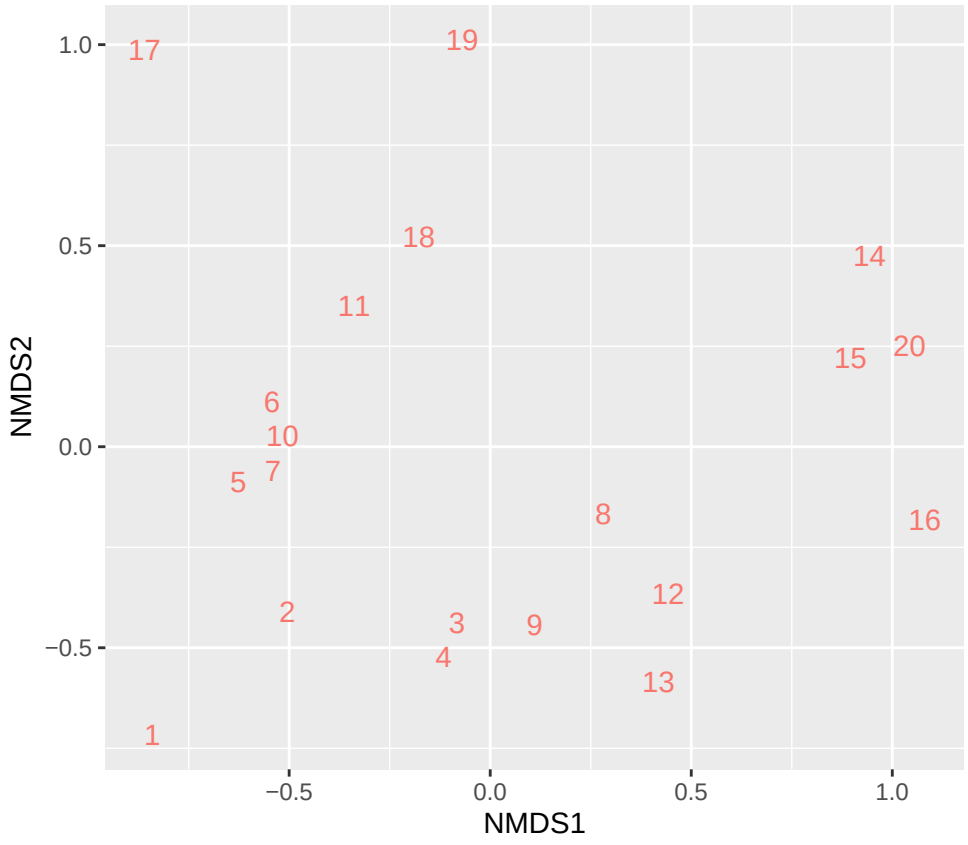
Output explained:

- **Call** - This is a reminder of how you fitted the NMDS.
- **Data** - This is the type of data used, species in this case.
- **Distance** - the dissimilarity metric, in this case the default is *Bray-Curtis*.
- **Dimensions** - the number of dimensions with the least stress.
- **Two convergent solutions found after 20 tries** - the NMDS did 20 random starts and found 2 solutions that were very similar to one another. If you ran the NMDS twice, it would say **. . . after 40 tries** here.

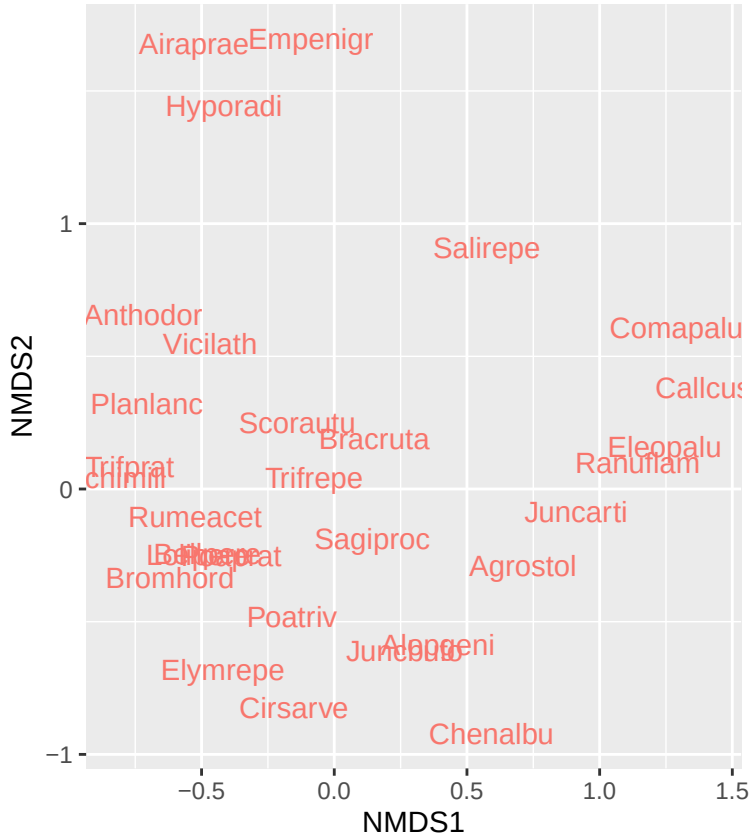
3.2 Visualise your NMDS results

Again you can use the `ordi_plot()` function to visualise the results. I usually find it easier to plot the samples and species separately, and compare the two graphs.

```
# Plot the NMDS sample (site) and attribute (species) scores
ordi_plot(dune_nmds, layers="sites", geom="text")
```



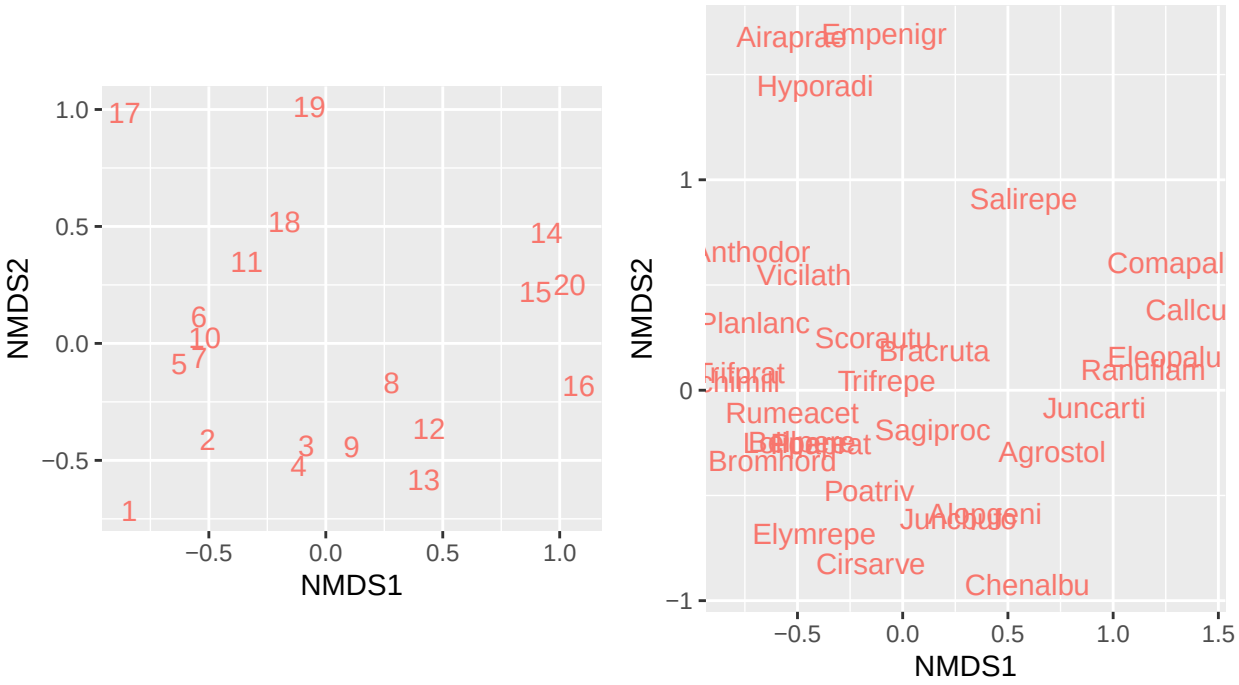
```
ordi_plot(dune_nmds, layers="species", geom="text")
```



If you wish to compare two graphs side-by-side, use the `multi_plot()` function from the `bio2020` package (type `?multi_plot` in the Console for help):

```
# Plot the NMDS sample (site) and attribute (species) scores but this
# time store the plots
dune_nmds_sites_plt <- ordi_plot(dune_nmds, layers="sites", geom="text")
dune_nmds_spp_plt   <- ordi_plot(dune_nmds, layers="species", geom="text")

# Use multi_plot to display side by side
multi_plot(dune_nmds_sites_plt, dune_nmds_spp_plt, cols=2)
```



Comparing between plots makes it easier to understand relationships. For example from these plots I can see that the species Elymrepe and Cirsarve are probably most common at site 1. Species Airaprae, Empenigr and Hyporadi are probably commonly found together. They are particularly characteristic of sites 17 and 19 which also have high NMDS2 scores.

Summary

Unconstrained methods provide useful ways of summarising tables of data from many different biological disciplines. They are worth considering if you have roughly 10 or more columns of data. This is particularly common in genomics and ecological studies, where multiple genes or species may be recorded for each sample or site. Doing lots of separate analyses on each column individually would be too time-consuming. These ordination methods also help you to identify **relationships** between variables, so that you can determine which samples (or sites) and which genes (or species) are most similar to each other.