Ordination

Fernando Racimo

adapted from notes by Marco Plebani and Steven Holland

2020-10-14

Contents

Chapter 1

Prerequisites

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

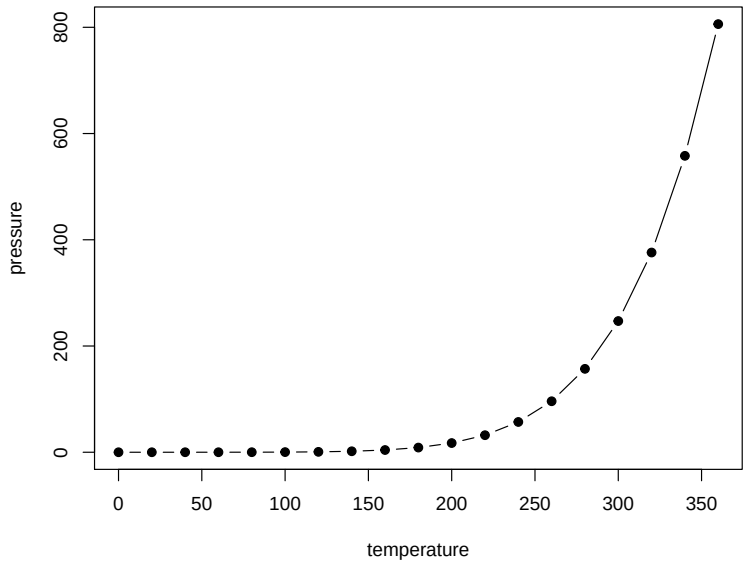


Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 3

Probability

We'll begin with probability.

Chapter 4

Linear Models

We describe linear models in this chapter.

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

Final words...

Chapter 7

Ordination

7.1 Libraries and Data

Today, we will work with the package vegan (useful for ordination techniques) and the packages ggplot2 and ggbiplot (useful for fancy plotting). Make sure all these libraries are installed before you begin. To install the package "ggbiplot", do the following:

install.packages(devtools)

library(devtools)

install_github("vqv/ggbiplot")

Let's begin by loading the necessary libraries:

```
library("vegan")
```

```
## Loading required package: permute
```

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

```
## This is vegan 2.5-6
```

```
library("ggplot2")
library("ggbiplot")
```

```
## Loading required package: plyr
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

We will use a dataset on measurements of particular parts of the iris plant, across individuals from three different species.

```
data(iris)
```

7.1.1 Exercise 1: Visualizing the data

Take a look at the iris data matrix. How many samples does it have? How many variables? What happens when you run the function plot() on this matrix? Which variables are most strongly correlated? (use the cor() function to answer this). Why do you think this could be?

7.2 Principal component analysis (PCA)

Perform a PCA of the data. The function prcomp() performs the PCA, and we can assign the result of this function to a new variable (let's call it "fit"). We must first remove the last column to whatever we give as input to prcomp, as the species names are a non-linear (categorical) variable and we don't have (for now) any natural measures of distance for species. The option scale=T standardizes the variables to the same relative scale, so that some variables do not become dominant just because of their large measurement unit.

```
fit<-prcomp(iris[-5], scale=TRUE)
```

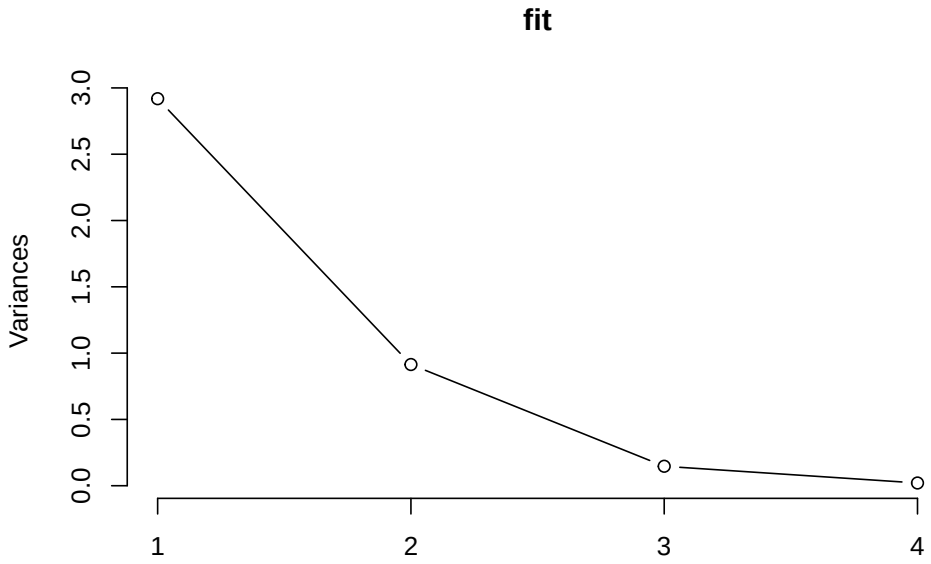
If we run the summary function on fit, it indicates that four PCs where created: the number of possible PCs always equals the number of original variables.

```
summary(fit)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

How much of the variance is explained by PC1? How much is explained by PC2?

```
plot(fit,type="lines")
```

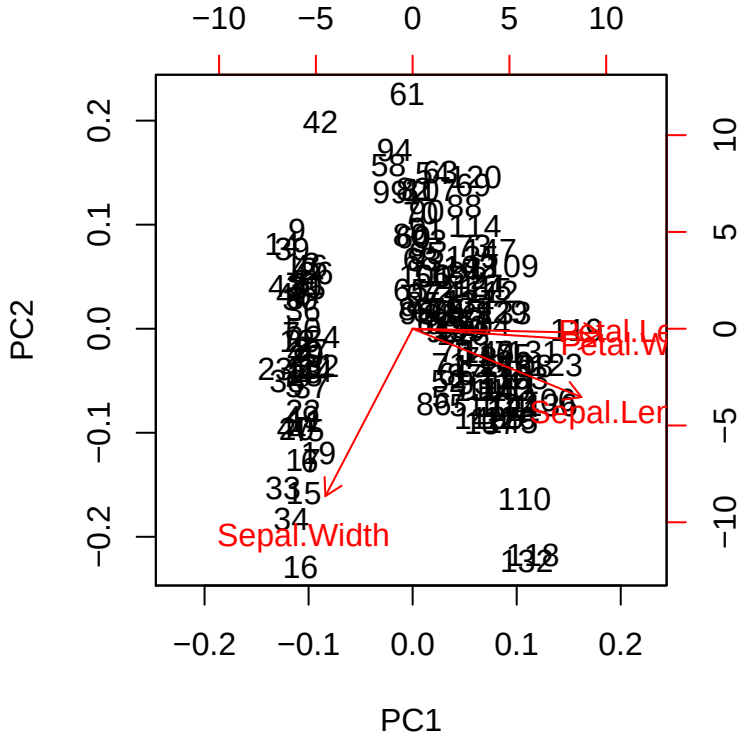


The "Rotation" matrix (fit[2]) contains the "loadings" of each of the original variables on the newly created PCs. Take a look at this matrix. The larger the absolute value of a variable in each PC, the more that variable contributes to that PC.

7.2.1 Biplots

We can use the function "biplot" to plot the first two PCs of our data. The plotted arrows provide a graphical rendition of the loadings of each of the original variables on the two PCs.

```
biplot(fit)
```



7.2.2 Exercise 2: Interpreting biplots

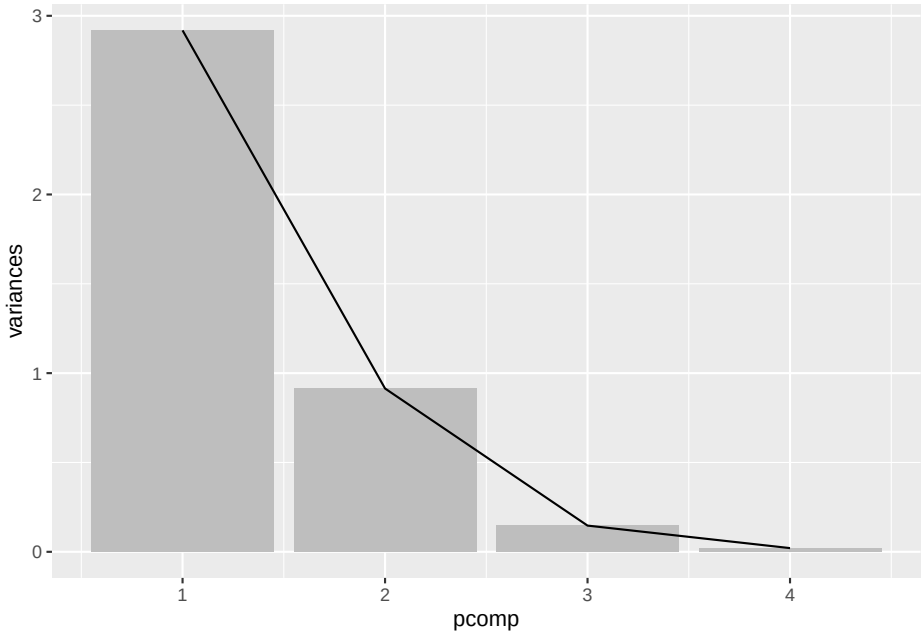
Across this reduced dimensional space, we can see that particular variables tend to co-vary quite strongly. Which ones? We can also see a separation into two groups on PC1. Which variables do you think would be most different between samples in one group and in the other?

7.2.3 PCA using ggplot

We can make prettier plots using ggplot2 and ggbiplot.

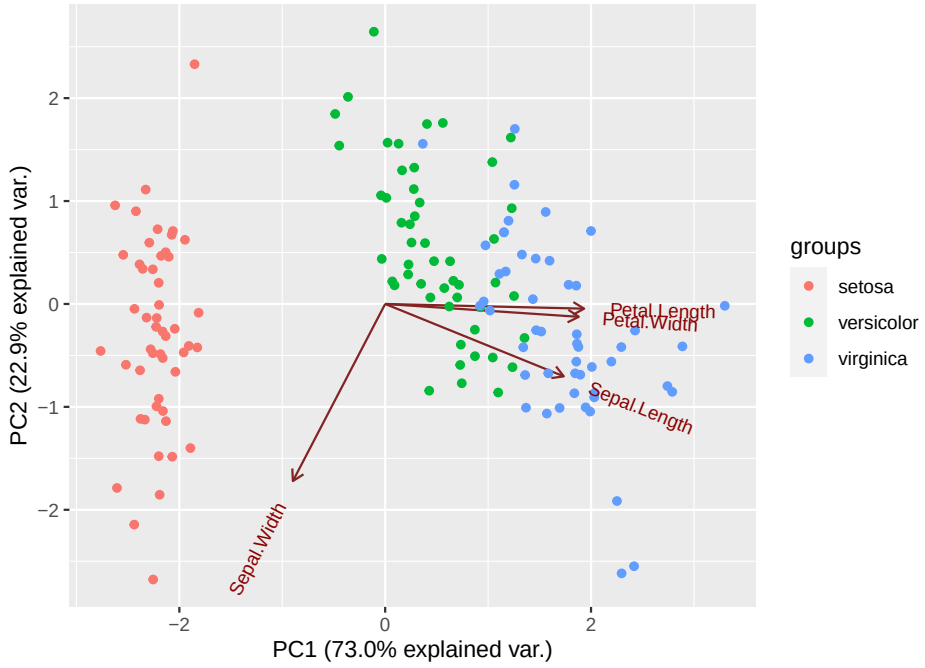
We first extract the variances of the principal components and then plot them:

```
variances <- data.frame(variances=fit$sdev**2, pcomp=1:length(fit$sdev))
varPlot <- ggplot(variances, aes(pcomp, variances)) + geom_bar(stat="identity", fill="
varPlot
```



We can also plot the first two PCs, like we had done before in base R, but now coloring the samples by their corresponding species. How are the species distributed along PC1?

```
Species<-iris$Species
iris_pca <- ggbiplot(fit,obs.scale = 1,
         var.scale=1,groups=Species,ellipse=F,circle=F,varname.size=3)
iris_pca
```



7.2.4 PCA under the hood

Rather than just using a ready-made function to compute a PCA, let's take a longer route to understand exactly what's happening under the hood of the prcomp() function.

First, let's standardize each column of our data so that each column has mean 0 and variance 1

```
irisdat <- iris[-5]
irisstandard <- apply(irisdat,2,function(x){(x-mean(x))/sd(x)})
```

Now, calculate the covariance matrix. Because the data has been standardized, this is equivalent to calculating the correlation matrix of the pre-standardized data.

```
cormat <- cov(irisstandard)
```

Then, extract the eigenvalues and eigenvectors of correlation matrix:

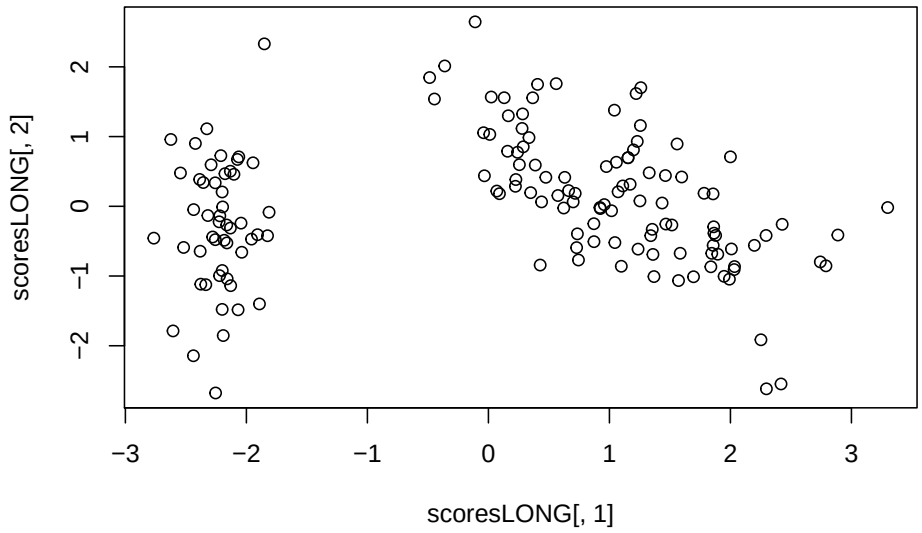
```
myEig <- eigen(cormat)
```

Now, we'll manually obtain certain values that were automatically computed by the prcomp function when we ran it earlier. We'll calculate the singular values (square root of eigenvalues) and also obtain the eigenvectors, also called loadings.

```
sdLONG <- sqrt(myEig$values)
loadingsLONG <- myEig$vectors
rownames(loadingsLONG) <- colnames(irisstandard)
```

Using the loadings, we can plot our original (standardized) data matrix into the new PC-space, by multiplying the data matrix by the matrix of loadings. Plotting the first two rows of the resulting product should reveal the location of our data points in the first two principal components (like we had before):

```
scoresLONG <- irisstandard %*% loadingsLONG
plot(scoresLONG[,1],scoresLONG[,2])
```



You can compare the results from the first section (using the ready-made function prcomp) and this section (taking a longer road), to check that the results are equivalent: the minimum and maximum differences in values for the loadings, the scores and standard deviations of the PCs are infinitesimally small.

```
range(fit$sdev - sdLONG)
```

```
## [1] -2.775558e-16  6.661338e-16
```

```
range(fit$rotation - loadingsLONG)
```

```
## [1] -7.216450e-16  3.330669e-16
```

```
range(fit$x - scoresLONG)
```

```
## [1] -2.865763e-15  2.220446e-15
```

7.3 NMDS

We'll now perform non-metric multidimensional scaling. Let's first take a look at the raw data we will use. This is a data matrix containing information about dune meadow vegetation. There are 30 species and 20 sites. Each cell corresponds to the number of specimens of a particular species that has been observed at a particular site (Jongman et al. 1987). As one can see, there are many sites where some species are completely absent (the cell value equals 0):

```
data(dune)
dune
```

```
##   Achimill Agrostol Airaprae Alopgeni Anthodor Bellpere Bromhord Chenalbu
## 1        1        0        0        0        0        0        0        0
## 2        3        0        0        2        0        3        4        0
## 3        0        4        0        7        0        2        0        0
## 4        0        8        0        2        0        2        3        0
## 5        2        0        0        0        4        2        2        0
## 6        2        0        0        0        3        0        0        0
## 7        2        0        0        0        2        0        2        0
## 8        0        4        0        5        0        0        0        0
## 9        0        3        0        3        0        0        0        0
## 10       4        0        0        0        4        2        4        0
## 11       0        0        0        0        0        0        0        0
## 12       0        4        0        8        0        0        0        0
## 13       0        5        0        5        0        0        0        1
## 14       0        4        0        0        0        0        0        0
## 15       0        4        0        0        0        0        0        0
## 16       0        7        0        4        0        0        0        0
## 17       2        0        2        0        4        0        0        0
## 18       0        0        0        0        0        2        0        0
## 19       0        0        3        0        4        0        0        0
## 20       0        5        0        0        0        0        0        0
##   Cirsarve Comapalu Eleopalu Elymrepe Empenigr Hyporadi Juncarti Juncbufo
## 1        0        0        0        4        0        0        0        0
## 2        0        0        0        4        0        0        0        0
## 3        0        0        0        4        0        0        0        0
## 4        2        0        0        4        0        0        0        0
## 5        0        0        0        4        0        0        0        0
## 6        0        0        0        0        0        0        0        0
## 7        0        0        0        0        0        0        0        2
## 8        0        0        4        0        0        0        4        0
## 9        0        0        0        6        0        0        4        4
## 10       0        0        0        0        0        0        0        0
## 11       0        0        0        0        0        2        0        0
## 12       0        0        0        0        0        0        0        4
```

```
## 13       0       0       0       0       0       0       0       3
## 14       0       2       4       0       0       0       0       0
## 15       0       2       5       0       0       0       3       0
## 16       0       0       8       0       0       0       3       0
## 17       0       0       0       0       0       2       0       0
## 18       0       0       0       0       0       0       0       0
## 19       0       0       0       0       2       5       0       0
## 20       0       0       4       0       0       0       4       0
##   Lolipere Planlanc Poaprat Poatriv Ranuflam Rumeacet Sagiproc Salirepe
## 1        7       0       4       2       0       0       0       0
## 2        5       0       4       7       0       0       0       0
## 3        6       0       5       6       0       0       0       0
## 4        5       0       4       5       0       0       5       0
## 5        2       5       2       6       0       5       0       0
## 6        6       5       3       4       0       6       0       0
## 7        6       5       4       5       0       3       0       0
## 8        4       0       4       4       2       0       2       0
## 9        2       0       4       5       0       2       2       0
## 10       6       3       4       4       0       0       0       0
## 11       7       3       4       0       0       0       2       0
## 12       0       0       0       4       0       2       4       0
## 13       0       0       2       9       2       0       2       0
## 14       0       0       0       0       2       0       0       0
## 15       0       0       0       0       2       0       0       0
## 16       0       0       0       2       2       0       0       0
## 17       0       2       1       0       0       0       0       0
## 18       2       3       3       0       0       0       0       3
## 19       0       0       0       0       0       0       3       3
## 20       0       0       0       0       4       0       0       5
##    Scorautu Trifprat Trifrepe Vicilath Bracruta Callcusp
## 1         0        0        0        0        0        0
## 2         5        0        5        0        0        0
## 3         2        0        2        0        2        0
## 4         2        0        1        0        2        0
## 5         3        2        2        0        2        0
## 6         3        5        5        0        6        0
## 7         3        2        2        0        2        0
## 8         3        0        2        0        2        0
## 9         2        0        3        0        2        0
## 10        3        0        6        1        2        0
## 11        5        0        3        2        4        0
## 12        2        0        3        0        4        0
## 13        2        0        2        0        0        0
## 14        2        0        6        0        0        4
## 15        2        0        1        0        4        0
## 16        0        0        0        0        4        3
```

```
## 17        2        0        0        0        0        0
## 18        5        0        2        1        6        0
## 19        6        0        2        0        3        0
## 20        2        0        0        0        4        3
```

Note that this data is non-linear, so our first instinct should not be to perform PCA on it. Because NMDS relies on "distances", we need to specify a distance metric that we'll use. The function for performing NMDS in the package 'vegan' is called metaMDS() and its default distance metric is "bray", which corresponds to the Bray-Curtis dissimilarity: a statistic used to quantify the compositional dissimilarity between two different sites, based on counts at each site

Let's perform NMDS ordination using the Bray-Curtis dissimilarity. Remember that, unlike PCA, NMDS requires us to specify the number of dimensions (k) a priori (the default in vegan is 2). It also performs a series of transformations on the data that are appropriate for ecological data (default: autotransform=TRUE). The trymax option ensures that the algorithm is started from different points (in our case, 50) to avoid local minima.

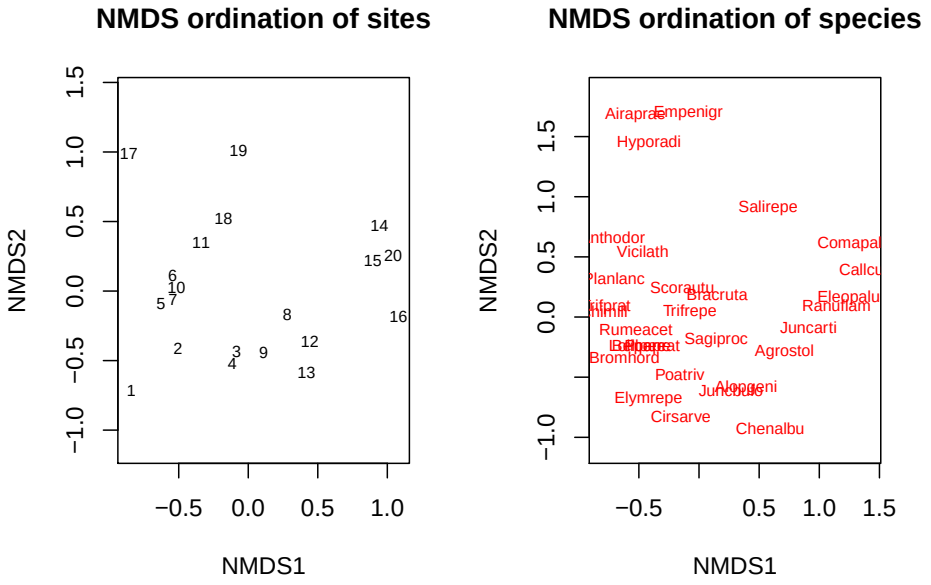
```
ord <- metaMDS(dune, k=2, autotransform = TRUE, trymax=50)
```

```
## Run 0 stress 0.1192678
## Run 1 stress 0.119268
## ... Procrustes: rmse 0.0002398743  max resid 0.0007343286
## ... Similar to previous best
## Run 2 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 0.02026095  max resid 0.06491316
## Run 3 stress 0.1192679
## Run 4 stress 0.1192679
## Run 5 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 7.631285e-05  max resid 0.0002364499
## ... Similar to previous best
## Run 6 stress 0.1192678
## Run 7 stress 0.1183186
## ... Procrustes: rmse 1.103049e-05  max resid 3.431477e-05
## ... Similar to previous best
## Run 8 stress 0.1183186
## ... Procrustes: rmse 2.829381e-05  max resid 9.144446e-05
## ... Similar to previous best
## Run 9 stress 0.1192682
## Run 10 stress 0.1192681
## Run 11 stress 0.1192685
## Run 12 stress 0.1808913
## Run 13 stress 0.1192678
## Run 14 stress 0.1886532
```

```
## Run 15 stress 0.1183186
## ... Procrustes: rmse 8.75512e-05  max resid 0.0002558773
## ... Similar to previous best
## Run 16 stress 0.2264808
## Run 17 stress 0.1183186
## ... Procrustes: rmse 5.289409e-05  max resid 0.0001699982
## ... Similar to previous best
## Run 18 stress 0.1886532
## Run 19 stress 0.1192682
## Run 20 stress 0.1192679
## *** Solution reached
```

As you can see, the function goes through a series of steps until convergence is reached. Let's plot the results:

```
par(mfrow=c(1,2))
plot(ord,display="sites",main="NMDS ordination of sites",type="t")
plot(ord,display="species",main="NMDS ordination of species",type="t")
```




```
par(mfrow=c(1,1))
```

7.3.1 Exercise 3: interpreting NMDS plots

What do these plots tell you about the distribution of species across sites? Which species tend to co-occcur with each other? Which sites tend to have similar species compositions?

Try changing the number of dimensions or the distance metric used. You can

take a look at the list of possible distances and their definitions using "?vegdist". Do the results change? Why?

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.21.