Multivariate Statistics

Prof. Richard Wilkinson

Spring 2021

Contents

Introduction

These lecture notes are now **complete**. I'll continue to correct typos but will not be adding additional material.

This module is concerned with the analysis of multivariate data, in which the response is a vector of random variables rather than a single random variable.

Part I of the module describes some basic concepts in Multivariate Analysis and then recaps and introduces some key ideas needed from linear algebra. Chapter 1 defines notation, introduces some datasets, and discusses exploratory data analysis. Chapter 2 provides a recap on some matrix algebra. Much of this will be familiar to you, but if not, we take the time to introduce the key mathematical concepts that will be relied upon during the module. Chapter 3 introduces matrix decompositions. We start with the spectral decomposition of square symmetric matrices (which you will have studied previously), and then introduce the singular value decomposition (SVD). The SVD is one of the most important concepts in this module, and is the key linear algebra technique behind many of the methods we will study.

A theme running through the module is that of dimension reduction. In Part II we consider three types of dimension reduction: Principal Components Analysis (in Chapter 4), whose purpose is to identify the main modes of variation in a multivariate dataset; Canonical Correlation Analysis (Chapter 5), whose purpose is to describe the association between two sets of variables; and Multidimensional Scaling (Chapter 6), in which the starting point is a set of pairwise distances, suitably defined, between the objects under study.

In Part III, we focus on methods of inference for multivariate data whose distribution is multivariate normal.

Finally, in Part IV, we focus on different methods of classification, i.e. allocating the observations in a sample to different subsets (or groups).

If you find any typos or mistakes, please email me at `r.d.wilkinson@nottingham.ac.uk` The notes have been significantly rewritten this year in order to adapt them for remote learning, and I am keen to fix as many of the mistakes as I can!

PART I: Prerequisites

Much of modern multivariate statistics (and machine learning) relies upon linear algebra. Consequently, we will spend some time reminding you of the basics of linear algebra (vector spaces, matrices etc), and introducing a few additional concepts that you may not have seen before. It is worth spending time familiarizing yourself with these ideas, as we will rely heavily upon this material in later chapters.

In Chapter 1 we explain what we mean by multivariate analysis and give some examples of multivariate data. We introduce basic definitions and concepts such as the sample covariance matrix, the sample correlation matrix and describe some simple exploratory data analysis techniques.

In Chapter 2 we summarise the definitions, ideas and results from matrix algebra that will be needed later in the module, most of which will be familiar to you. In particular, we will introduce vector spaces and the concept of a basis for a vector space, discuss the column, row and null space of matrices, and discuss inner product spaces and projections. We also define the centering matrix.

In Chapter 3 we recap the eigen or spectral decomposition of square symmetric matrices, and introduce the singular value decomposition (SVD) which generalises the concept of eigenvalues for non-square matrices. We will rely upon this material inlater chapters.

Chapter 1

Statistical Preliminaries

In this chapter we will define some notation, and recap some basic statistical properties and results.

There are recorded videos for the following topics in this chapter:

- Notation and datasets
- Exploratory data analysis
- Random vectors

1.1 Notation

We will think of datasets as consisting of measurements of p different **variables** for n different **cases/subjects**. We organise the data into an $n \times p$ **data matrix**.

Multivariate analysis (MVA) refers to data analysis methods where there are two or more **response** variables for each case (you are familiar with situations where there is more than one explanatory variable, e.g., multiple linear regression).

We shall often write the data matrix as $\mathbf{X}$ $(n \times p)$ where

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1^\top & - \\ - & \mathbf{x}_2^\top & - \\ - & .. & - \\ - & \mathbf{x}_n^\top & - \end{bmatrix}$$

The vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ are the observation vectors for each of the n subjects.

- The n *rows* of $\mathbf{X}$ are $\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top$ - each row contains the p observations on a single subject.
- The p columns of $\mathbf{X}$ correspond to the p variables being measured, i.e., they contain the measurements of the same variable across all n subjects.

Important remark on notation: Throughout the module we shall use

- non-bold letters, whether upper or lower case, to indicate scalar (i.e. real-valued) quantities, e.g., x, y
- lower-case letters in bold to signify column vectors, e.g., $\mathbf{x}, \mathbf{y}$
- upper case letters in bold to signify matrices, e.g., $\mathbf{X}, \mathbf{Y}$.

This convention for bold letters will also apply to random quantities. So, in particular, for a random vector we always use (bold) lower case, and for a random matrix we always use bold upper-case, regardless of whether we are referring to (i) the unobserved random quantity or (ii) its observed value. It should always be clear from the context which of these two interpretations (i) or (ii) is appropriate.

1.1.1 Example datasets

Example 1.1. The football league table is an example of multivariate data. Here W = number of wins, D = number of draws, F = number of goals scored and A = number of goals conceded for four teams. In this example we have $p = 4$ variables $(W, D, F, A)^\top$ measured on $n = 4$ cases (teams).

Team	W	D	F	A
USA	1	2	4	3
England	1	2	2	1
Slovenia	1	1	3	3
Algeria	0	1	0	2

The data vector for the USA is

$$x_1^\top = (1, 2, 4, 3)$$

Example 1.2. Exam marks for a set of n students where P = mark in probability and S = mark in statistics. Let x_{ij} denote the jth variable measured on the ith subject.

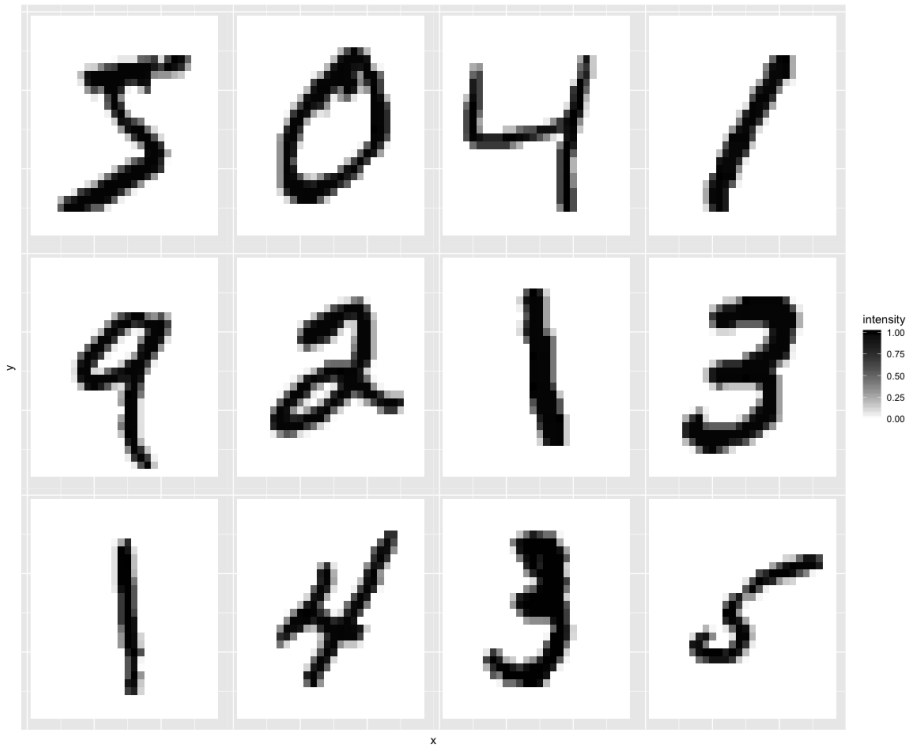
Student	P	S
1	x_{11}	x_{12}
2	x_{21}	x_{22}
$\vdots$	$\vdots$	$\vdots$
n	x_{n1}	x_{n2}

Example 1.3. The `iris` dataset is a famous set of measurements collected on the sepal length and width, and the petal length and width, of 50 flowers for each of 3 species of iris (setosa, versicolor, and virginica). The dataset is built

into R (try typing `iris` in R) and is often used to demonstrate multivariate statistical methods. For these data, $p = 5$, and $n = 150$.

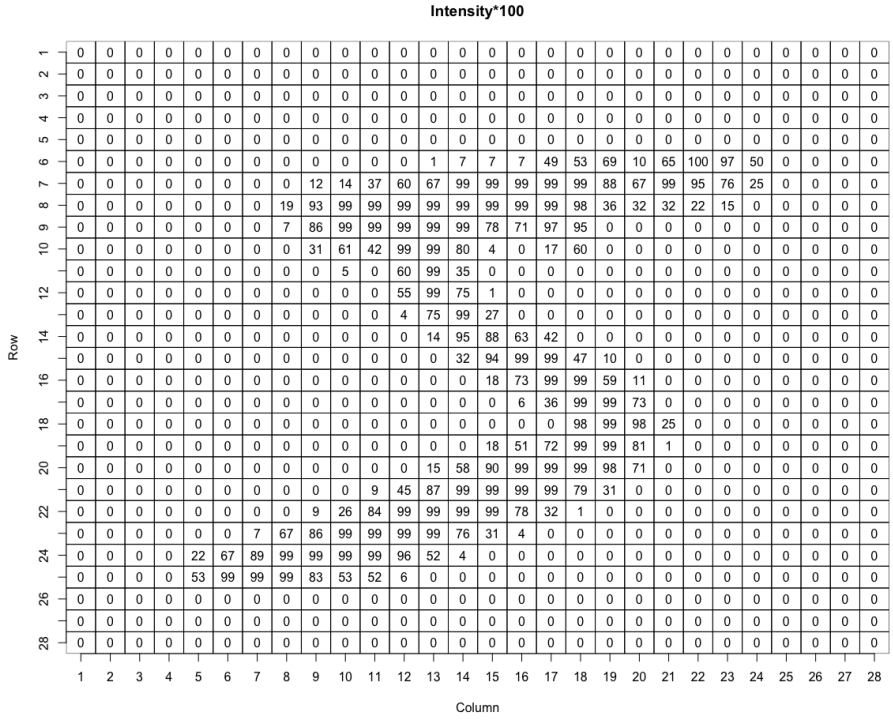
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica

Example 1.4. The MNIST dataset is a collection of handwritten digits that is widely used in statistics and machine learning to test algorithms. It contains 60,000 images of hand-written digits. Here are the first 12 images:



Each digit has been converted to a grid of 28×28 pixels, with a grayscale intensity level specified for each pixel. When we store these on a computer, we

flatten each grid to a vector of length 784. So for this dataset, $n = 60,000$ and $p = 784$. As an example of what the data look like, the intensities (times 100) for the first image above are shown in the plot below:



1.1.2 Aims of multivariate data analysis

The aim of multivariate statistical analysis is to answer questions such as:

- How can we visualise the data?
- What is the joint distribution of marks?
- Can we simplify the data? For example, we rank football teams using $3W + D$ and we rank students by their average module mark. Is this fair? Can we reduce the dimension in a better way?
- Can we use the data to discriminate, for example, between male and female students?
- Are the different iris species different shapes?
- Can we build a model to predict the intended digit from an image of someones handwriting? Or predict the species of iris from measurements of its sepal and petal?

We could just apply standard univariate techniques to each variable in turn, but this ignores possible dependencies between the variables which we must represent

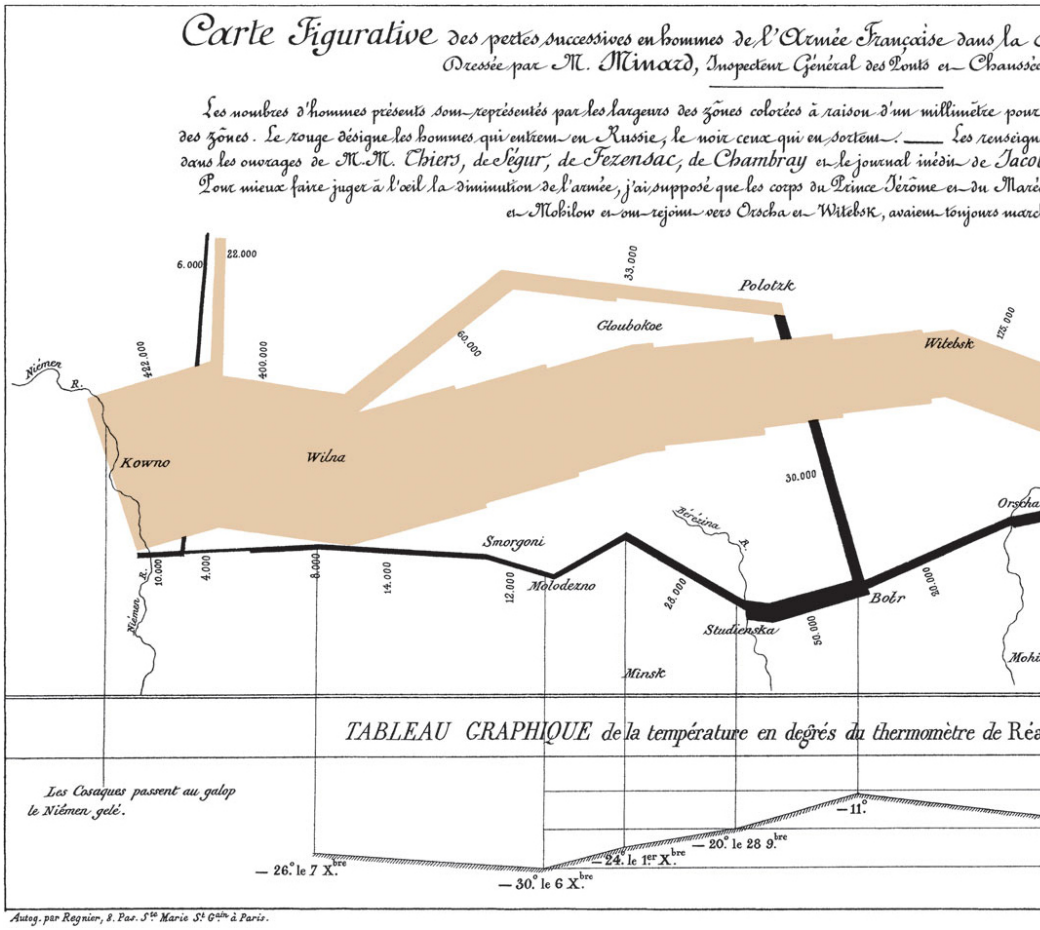
to draw valid conclusions.

What is the difference between MVA and standard linear regression?

- In standard linear regression we have a scalar response variable, y say, and a vector of covariates, $\mathbf{x}$, say. The focus of interest is on how knowledge of $\mathbf{x}$ influences the distribution of y (in particular, the mean of y). In contrast, in MVA the focus is a vector $\mathbf{y}$, in which all the components of $\mathbf{y}$ are viewed as responses rather than covariates, possibly with additional covariate information $\mathbf{x}$. We will discuss this further in Chapter 10.

1.2 Exploratory data analysis (EDA)

A picture is worth a thousand words



\begin{figure}

\caption{Charles Joseph Minard's famous map of Napoleon's 1812 invasion of Russian. It displays six types of data in two dimensions.} \end{figure}

Before trying any form of statistical analysis, it is always a good idea to do some form of exploratory data analysis to understand the challenges presented by the data. As a minimum, this usually involves finding out whether each variable is continuous, discrete, or categorical, doing some basic visualization (plots), and perhaps computing a few summary statistics such as the mean and variance.

1.2.1 Data visualization

Visualising datasets before fitting any models can be extremely useful. It allows us to see obvious patterns and relationships,and may suggest a sensible form of analysis. With multivariate data, finding the right kind of plot is not always simple, and many different approaches have been proposed.

When $p = 1$ or $p = 2$ we can simply draw histograms and scatter plots (respectively) to view the distribution. For $p \geq 3$ the task is harder. One solution is a matrix of pair-wise scatter plots using the `pairs` command in R. The graph below shows the relationship between goals scored (F), goals against (A) and points (PT) for 20 teams during a recent Premiership season.

We can instantly see that points and goals scored are positively correlated, and that points and goals conceded (A) are negatively correlated (this is not a surprise of course).

R has a good basic plotting functionality. However, we will sometimes use packages that provide additional functionality. The first time you use a package you may need to install it. We can use `ggplot2` and `GGally` (which adds functionality to ggplot2) to add colour and detail to pairs plots. For example

```
data(iris)
library(ggplot2)
library(GGally)
# pairs(iris) # - try the pairs command for comparison
ggpairs(iris, columns=1:4,  mapping=ggplot2::aes(colour = Species),
        upper = list(continuous = wrap("cor", size = 3)))  # fix the font size
```

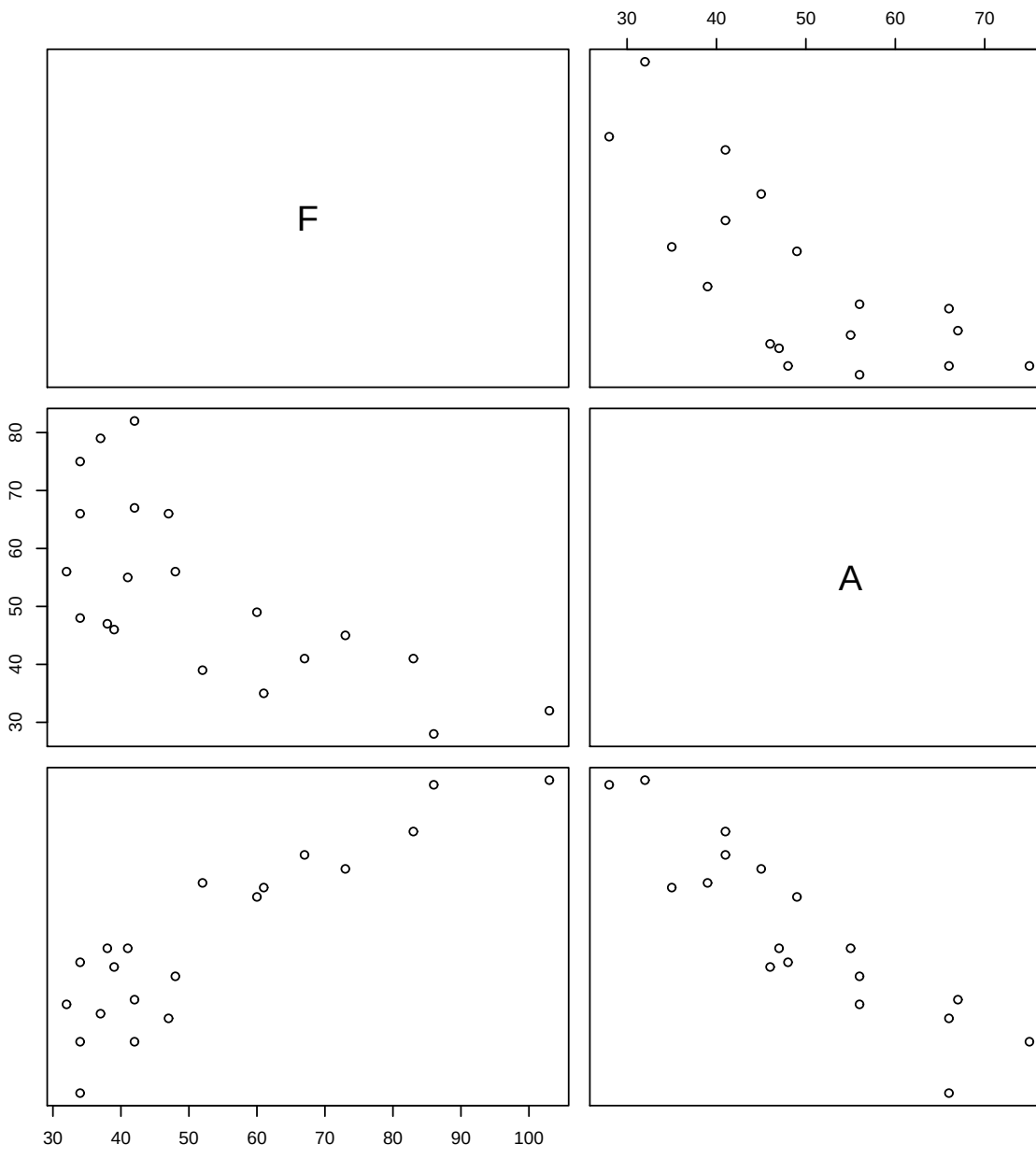
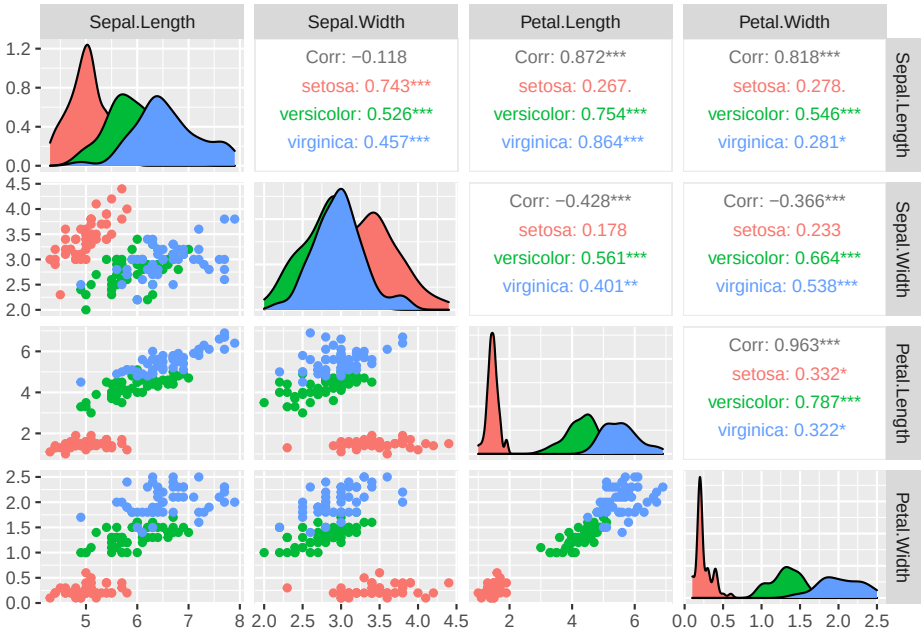


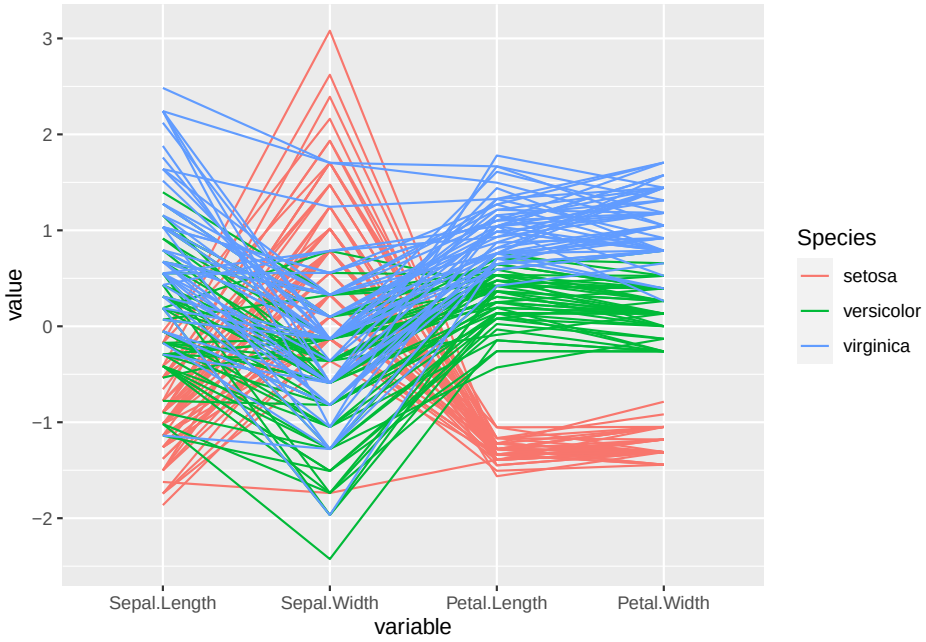
Figure 1.1: Scatter plots of goals for (F), goals against (A) and points (PT) for a recent Premier League Season



This plot allows us to instantly see that there are clear differences between the three species of iris, at least when we look at the pairs plots. The benefit of adding colour in this case is that we can see the differences between the different species. Note how the sepal length and width are (weakly) negatively correlated across the entire dataset, but are positively correlated when we look at a single species at a time. We would have missed this information if we only used the `pairs` command (try it!).

Note that it is possible to miss key relationships when looking at *marginals* plots such as these, as they only show two variables at a time. More complex relationships between three or more variables will not be visible. It is difficult visualize data in three or more dimensions. Many different types of plot have been proposed (e.g. Google Chernoff faces). One approach is to use a *parallel line* plot

```
ggparcoord(iris, 1:4, groupColumn=5)
```



Each case is represented by a single line, and here we have the information shown for the four continuous variables. The fifth variable `Species` is a discrete factor, and is shown by colouring the lines.

If you not familiar with `ggplot2`, a nice introduction can be found here. Details about 'GGally can be found here. A good way to see the variety of plots that are possible, and to find code to create them, is to browse plot galleries such as those available here and here.

1.2.2 Summary statistics

It is often useful to report a small number of numerical summaries of the data. In univariate statistics we define the sample mean and sample variance of samples $x_1, \ldots, x_n$ to be

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad s_{xx} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

and for two samples, $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, we define the sample covariance to be

$$s_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}).$$

Analogous multivariate quantities can be defined as follows:

Definition 1.1. For a sample of n points, each containing p variables, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^p$, the **sample mean** and **sample covariance matrix** are

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \tag{1.1}$$

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \tag{1.2}$$

where $\mathbf{x}_i \in \mathbb{R}^p$ denotes the p variables observed on the ith subject.

Note that

- $\bar{\mathbf{x}} \in \mathbb{R}^p$. The jth entry in $\bar{\mathbf{x}}$ is simply the (univariate) sample mean of the jth variable.
- $\mathbf{S} \in \mathbb{R}^{p \times p}$. Note that the ij^{th} entry of $\mathbf{S}$ is s_{ij}, the sample covariance between variable i and variable j. The i^{th} diagonal element is the (univariate) sample variance of the ith variable.

- $\mathbf{S}$ is symmetric since $s_{ij} = s_{ji}$.
- an alternative formula for $\mathbf{S}$ is

$$\mathbf{S} = \frac{1}{n}\left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right) - \bar{\mathbf{x}}\bar{\mathbf{x}}^\top.$$

- We have divided by n rather than $n-1$ here, which gives the maximum likelihood estimator of the variance, rather than the unbiased variance estimator that is often used.

Definition 1.2. The **sample correlation matrix**, $\mathbf{R}$, is the matrix with ij^{th} entry r_{ij} equal to the sample correlation between variables i and j, that is

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}.$$

Note that

- If $\mathbf{D} = \text{diag}(\sqrt{s_{11}}, \ldots, \sqrt{s_{pp}})$, then

$$\mathbf{R} = \mathbf{D}^{-1}\mathbf{S}\mathbf{D}^{-1}$$

- $\mathbf{R}$ is symmetric
- the diagonal entries of $\mathbf{R}$ are exactly 1 (each variable is perfectly correlated with itself)
- $|r_{ij}| \leq 1$ for all i, j

Note that if we change the unit of measurement for the $\mathbf{x}_i$'s then $\mathbf{S}$ will change but $\mathbf{R}$ will not.

Definition 1.3. The **total variation** in a data set is usually measured by $\text{tr}(\mathbf{S})$ where tr() is the trace function that sums the diagonal elements of the matrix. That is,

$$\text{tr}(\mathbf{S}) = s_{11} + s_{22} + \ldots + s_{pp}.$$

In other words, it is the sum of the univariate variances of each of the p variables.

1.3 Random vectors and matrices

Definition 1.4. The **population mean vector** of the random vector $\mathbf{x}$ is

$$\boldsymbol{\mu} = \mathbb{E}(\mathbf{x}).$$

The **population covariance matrix** of $\mathbf{x}$ is

$$\boldsymbol{\Sigma} = \mathbb{V}\,\text{ar}(\mathbf{x}) = \mathbb{E}\left((\mathbf{x} - \mathbb{E}(\mathbf{x}))(\mathbf{x} - \mathbb{E}(\mathbf{x}))^\top\right).$$

The **covariance** between $\mathbf{x}$ $(p \times 1)$ and $\mathbf{y}$ $(q \times 1)$ is

$$\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left((\mathbf{x} - \mathbb{E}(\mathbf{x}))(\mathbf{y} - \mathbb{E}(\mathbf{y}))^\top\right).$$

Let $\mathbf{A}$ denote a $q \times p$ constant matrix, and let $\mathbf{b}$ a constant vector of size $q \times 1$. Expectation is a linear operator in the sense that

$$\mathbb{E}(\mathbf{A}\mathbf{x} + \mathbf{b}) = \mathbf{A}\mathbb{E}(\mathbf{x}) + \mathbf{b} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}.$$

The following properties follow:

- $\mathbb{V}\,\text{ar}(\mathbf{x}) = \mathbb{E}(\mathbf{x}\mathbf{x}^\top) - \boldsymbol{\mu}\boldsymbol{\mu}^\top$.
- $\mathbb{V}\,\text{ar}(\mathbf{A}\mathbf{x} + \mathbf{b}) = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top$
- $\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y}) = \mathbb{E}(\mathbf{x}\mathbf{y}^\top) - \mathbb{E}(\mathbf{x})\mathbb{E}(\mathbf{y})^\top$.
- $\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{x}) = \boldsymbol{\Sigma}$.
- $\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y}) = \mathbb{C}\,\text{ov}(\mathbf{y}, \mathbf{x})^\top$.
- $\mathbb{C}\,\text{ov}(\mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{y}) = \mathbf{A}\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y})\mathbf{B}^\top$
- If $p = q$ then

$$\mathbb{V}\,\text{ar}(\mathbf{x} + \mathbf{y}) = \mathbb{V}\,\text{ar}(\mathbf{x}) + \mathbb{V}\,\text{ar}(\mathbf{y}) + \mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y}) + \mathbb{C}\,\text{ov}(\mathbf{y}, \mathbf{x}).$$

Finally, note that if $\mathbf{x}$ and $\mathbf{y}$ are independent (in which case I will write $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$) then $\mathbb{C}\,\text{ov}(\mathbf{x}, \mathbf{y}) = \mathbf{0}_{p,q}$, i.e., a $p \times q$ matrix of zeros.

1.3.1 Estimators

The population mean vector $\boldsymbol{\mu}$ and population covariance matrix $\boldsymbol{\Sigma}$ will usually be unknown. We can use data to **estimate** these quantities.

- The sample mean $\bar{\mathbf{x}}$ is often used as an estimator of $\boldsymbol{\mu}$.

- The sample covariance matrix $\mathbf{S}$ is often used as an estimator of $\mathbf{\Sigma}$.

Equation (1.1) gives an unbiased estimator of the sample mean. The sample covariance matrix (1.2) is a biased estimator of the population covariance matrix. An unbiased estimate is obtained by dividing by $n-1$ rather than n in Equation (1.2).

1.4 Computer tasks

If you haven't done so already, please download and install R and Rstudio. R is the programming language, and Rstudio is an integrated development environment that makes using R much more pleasurable. My advice is to always use Rstudio and never run code in R itself.

0. **For complete beginners:** For those who are completely new to R (or those who want a refresher), I recommend working through an online tutorial. This tutorial looks good, but contains more than you'll need.

1. **Warm-up:** The most important aspects of R to focus on for this module are

   - Basic plotting
   - Manipulation of matrices and data frames.

   Let's look at the `iris` dataset.

   - Can you plot the sepal length against the sepal width?

   We'll now do some exercises on data manipulation. Note that there are several ways to do basic data manipulation in R. You can use base R commands or if you prefer, you can use the dplyr commands which are part of the `tidyverse` packages. For example, to select columns, in base you can do:

```
iris[,2] # selects column 2
iris$Sepal.Width # selects the same column by name
```

or using dplyr you can do

```
library(dplyr)
select(iris, "Sepal.Width")
```

   - Can you select the column of the `iris` data that contains just the sepal length and add it to the sepal width?

   To select only certain rows of the data (i.e. to filter it), we can again use either base R or dplyr.

```
iris[iris[,3]<5,] # select all rows that have a petal length less than 5.
filter(iris, Petal.Length<5) # do the same thing using dplyr
```

- Can you now select all the rows of the `iris` data frame that are for species *setosa*? What is the mean petal length for these flowers?
- Can you select all the flowers that have a sepal length greater than 5? What is the proportion of each species of iris in this set?

A nice aspect of dplyr is that you can chain commmands together. So for example, to select the versicolour flowers with petal width less than 1.5, we can do

```
iris %>% filter(Species=='versicolor') %>% filter(Petal.Width<1.5)
```

- Can you select all the flowers that have a sepal length greater than 6, and a petal length less than 5? What is the proportion of each species in this set?

Note that `iris` is a data frame

```
is.data.frame(iris)
```

```
## [1] TRUE
```

which is a type of structure used in R. This is convenient for some tasks, but not for others. Let's first extract the four numerical columns and store them as a matrix X.

```
is.matrix(iris)
```

```
## [1] FALSE
```

```
X <- as.matrix(iris[,1:4])
is.matrix(X)
```

```
## [1] TRUE
```

- Select the 4 numerical columns and multiply the first column by 1, the second by 2, the third by 3, and the 4th by 4. One way to do this is by multiplying X by the diagonal matrix

```
diag(1:4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    2    0    0
## [3,]    0    0    3    0
## [4,]    0    0    0    4
```

2. The table below shows the module marks for 5 students on the modules G11PRB (P) and G11STA (S).

Student	P	S
A	41	63
B	72	82
C	46	38
D	77	57
E	59	85

- As an exercise, calculate the sample mean, sample covariance, sample correlation and total variation by hand.
- Now calculate these in R using `colMeans`, `cov`, and `cor`. These commands assume each column is a different variable, and each row a different observation.

```
library(dplyr)
Ex1 <- data.frame(
  Student=LETTERS[1:5],
  P = c(41,72,46,77,59),
  S = c(63,82,38,57,85)
  )

Ex1 %>% select_if(is.numeric) %>% colMeans
```

```
##  P  S
## 59 65
```

```
Ex1 %>% select_if(is.numeric) %>% cov
```

```
##       P     S
## P 246.5 116.0
## S 116.0 371.5
```

Note that by default R uses $n-1$ in the denominator for the variance and covariance commands, whereas we used n in our definition.

We will be using the `dplyr` R package to perform basic data manipulation in R. If you are unfamiliar with `dplyr`, you can read about it at https://dplyr.tidyverse.org/. The pipe command `%>%` is particularly useful for chaining together multiple commands.

You could compute the same quantities using more familiar commands by selecting the numerical columns:

```
colMeans(Ex1[,2:3])
```

```
##  P  S
## 59 65
```

```
cov(Ex1[,2:3])
```

```
##       P     S
## P 246.5 116.0
## S 116.0 371.5
```

- Can you compute the covariance matrix using the definition in Equation (1.2)?

3. The `mtcars` dataset is another built-in dataset in R. You can read about it by typing `?mtcars` in R. Note that some of the variables are factors. You can ensure R treats them as factors by using the following command to create a dataset where they are listed as factors:

```
mtcars2 <- within(mtcars, {
   vs <- factor(vs, labels = c("V", "S"))
   am <- factor(am, labels = c("automatic", "manual"))
   cyl  <- ordered(cyl)
   gear <- ordered(gear)
   carb <- ordered(carb)
})
```

`Work with the `mtcars2` dataframe when you use `ggplot2`.`

- Create some plots to explore the structure of this dataset using ggplot2.
- Try using the `pairs` command from base R and the `ggpairs` command from GGally.
- Try colouring the scatter plots accoring to whether the car is automatic or not. - Create another plot using colour to represent the number of gears.
- Find another type of plot from one of the plot galleries and try to create a similar plot with these data.

4. We can 100 generate samples from the multivariate normal distribution with mean vector

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

as follows (you may need to install the R package `mvtnorm` first):

```
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.6.2
```

```
mu = c(1,0)
Sigma=matrix(c(2,1,1,2), nr=2)
X <- rmvnorm(n=100, mean=mu, sigma=Sigma)
```

- Compute the sample mean and covariance matrix of these samples.
- Generate a new sample dataset, X, and recompute the sample mean and covariance matrix. What do you notice?
- Try changing n, the number of samples (making it much larger say), and now recomputing the mean and covariance. What do you notice?

5. **Optional** Download the MNIST data from Moodle and load it into R.

```
load('mnist.rda')
```

This loads a list `mnist` that splits the data into two parts

```
mnist$train ## a training set of 60000 images
mnist$test ## a test set of 10000 images
```

Let's just look at the training set. This is also a list containing the image intensities and the image labels

```
mnist$train$x # image intensities
mnist$train$y # image labels
```

If we select just the first image we can see it is a vector of length 784 containing numbers between 0 and 1.

```
mnist$train$x[1,]
```

I've created a function to help you plot these images.

```
library(reshape2)
library(ggplot2)


plot.mnist <- function(im){
  #im[im<0]<-0 # set any negative intensities to zero
  #im[im>1]<-1 # set an intensities bigger than 1 to 1.


  if(is.vector(im)){ # a single image

    A<-matrix(im, nr=28, byrow=F)
    C<- melt(A, varnames = c("x", "y"), value.name = "intensity")
    p<-ggplot(C, aes(x = x, y = y, fill = intensity))+
      geom_tile(aes(fill=intensity))+
      scale_fill_gradient(low='white', high='black')+
      scale_y_reverse()+theme(
        strip.background = element_blank(),
        strip.text.x = element_blank(),
        panel.spacing = unit(0, "lines"),
        axis.text = element_blank(),
        axis.ticks = element_blank()
```

```
      )
  }
  else{
    if (dim(im)[2]!=784){
      im = t(im)
    }
    n <- dim(im)[1]
    As <- array(im, dim = c(n, 28, 28))

    Cs<- melt(As, varnames = c("image","x", "y"), value.name = "intensity")
    p<-ggplot(Cs, aes(x = x, y = y, fill = intensity))+
      geom_tile(aes(fill=intensity))+
      scale_fill_gradient(low='white', high='black')+
      facet_wrap(~ image, nrow = floor(sqrt(n))+1, ncol = floor(sqrt(n))+1)+
      scale_y_reverse()+theme(
        strip.background = element_blank(),
        strip.text.x = element_blank(),
        panel.spacing = unit(0, "lines"),
        axis.text = element_blank(),
        axis.ticks = element_blank()
      )

  }
  return(p)
}
```

- Use this command to plot the first 10 images from the MNIST training set.
- Select all the 5s from the MNIST training set. Plot a selection of these digits.

1.5 Exercises

1. Show that the two formulae for the population covariance matrix $\mathbf{\Sigma}$ are equivalent, i.e. show that

$$\mathbf{\Sigma} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top] = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top.$$

2. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a p-dimensional sample with mean $\bar{\mathbf{x}}$ and sample covariance matrix $\mathbf{S}$. Consider the transformation $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{c}$ where $\mathbf{A}$ is a fixed $q \times p$ matrix and $\mathbf{c}$ is a fixed q-dimensional vector. Let $\mathbf{T}$ be the sample covariance matrix of $\mathbf{y}_1, \ldots, \mathbf{y}_n$. Show

- $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{c}$,

- $\mathbf{T} = \mathbf{A}\mathbf{S}\mathbf{A}^\top$.

  Assuming now that $\mathbf{x}$ is a random vector with $\mathbb{E}(\mathbf{x}) = \boldsymbol{\mu}$, $\mathbb{V}\mathrm{ar}(\mathbf{x}) = \boldsymbol{\Sigma}$, $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{c}$ with $\mathbf{A}$ and $\mathbf{c}$ as before, $\mathbb{E}(\mathbf{y}) = \boldsymbol{\phi}$ and $\mathbb{V}\mathrm{ar}(\mathbf{y}) = \boldsymbol{\Omega}$, what are the population analogues of the results above?

3. A sample of size $n = 144$ produced the following summary statistics

$$\sum_{i=1}^{n} \mathbf{x}_i = \begin{pmatrix} 392.2 \\ 1530.8 \end{pmatrix} \qquad \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top = \begin{pmatrix} 1101.88 & 4305.17 \\ 4305.17 & 17120.88 \end{pmatrix}.$$

   Calculate the sample mean, the sample covariance matrix and the sample correlation coefficient.

4. Let $\mathbf{x}$ and $\mathbf{y}$ be independent random p-dimensional vectors. Assuming that all relevant moments exist, show that for any real scalars α and β,

$$\mathbb{V}\mathrm{ar}(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha^2 \mathbb{V}\mathrm{ar}(\mathbf{x}) + \beta^2 \mathbb{V}\mathrm{ar}(\mathbf{y}).$$

   What is the corresponding formula when $\mathbf{x}$ and $\mathbf{y}$ are not independent? Express your answer in terms of $\mathbb{V}\mathrm{ar}(\mathbf{x})$, $\mathbb{V}\mathrm{ar}(\mathbf{y})$ and $\mathbb{C}\mathrm{ov}(\mathbf{x}, \mathbf{y})$.

Chapter 2

Review of linear algebra

Modern statistics and machine learning rely heavily upon linear algebra, nowhere more so than in multivariate statistics. In the first part of this chapter (sections 2.1 and 2.2) we review some concepts from linear algebra that will be needed throughout the module, including vector spaces, row and column spaces, the rank of a matrix, etc. Hopefully most of this will be familiar to you.

We then cover some basic details on inner-product or normed spaces in 2.3, which are vector spaces equipped with a concept of distance and angle. Finally, in Section 2.4 we will describe the centering matrix. Further details and proofs for this section will be tackled in the exercises in Section 2.6.

I do not provide proofs for many of the results stated in this chapter, but instead prove a small selection which I think it is useful to see. For a complete treatment of the linear algebra needed for this module, see the excellent book "Linear algebra and learning from data" by Gilbert Strang.

I have recorded videos on some (but not all) of the topics in these notes:

- Vector spaces
- Matrices
- Inner product spaces
- Orthogonal matrices
- Projection matrices

2.1 Basics

In this section, we recap some basic definitions and notation. Hopefully this material will largely be familiar to you.

2.1.1 Notation

The matrix $\mathbf{A}$ will be referred to in the following equivalent ways:

$$\begin{aligned}
\mathbf{A} = \overset{n\times p}{\mathbf{A}} &= \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{np} \end{pmatrix} \\
&= [a_{ij} : i = 1, \dots, n; j = 1, \dots, p] \\
&= (a_{ij}) \\
&= \begin{pmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_n^\top \end{pmatrix}
\end{aligned}$$

where the a_{ij} are the individual entries, and $\mathbf{a}_i^\top = (a_{i1}, a_{i2}, \dots, a_{ip})$ is the i^{th} row.

A matrix of order 1×1 is called a *scalar*.

A matrix of order $n \times 1$ is called a *(column) vector*.

A matrix of order $1 \times p$ is called a *(row) vector*.

e.g. $\overset{n\times 1}{\mathbf{a}} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$ is a column vector.

The $n \times n$ *identity matrix* $\mathbf{I}_n$ has diagonal elements equal to 1 and off-diagonal elements equal to zero.

A *diagonal* matrix is an $n \times n$ matrix whose off-diagonal elements are zero. Sometimes we denote a diagonal matrix by $\text{diag}\{a_1, \dots, a_n\}$.

$$\mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{diag}\{1, 2, 3\} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

2.1.2 Elementary matrix operations

1. *Addition/Subtraction.* If $\overset{n\times p}{\mathbf{A}} = [a_{ij}]$ and $\overset{n\times p}{\mathbf{B}} = [b_{ij}]$ are given matrices then

$$\mathbf{A} + \mathbf{B} = [a_{ij} + b_{ij}] \qquad \text{and} \qquad \mathbf{A} - \mathbf{B} = [a_{ij} - b_{ij}].$$

2. *Scalar Multiplication.* If λ is a scalar and $\mathbf{A} = [a_{ij}]$ then

$$\lambda\mathbf{A} = [\lambda a_{ij}].$$

3. *Matrix Multiplication.* If $\overset{n\times p}{\mathbf{A}}$ and $\overset{p\times q}{\mathbf{B}}$ are matrices then $\mathbf{AB} = \overset{n\times q}{\mathbf{C}} = [c_{ij}]$ where
$$c_{ij} = \sum_{k=1}^{p} a_{ik}b_{kj}, \qquad i = 1, \ldots, n, \qquad j = 1, \ldots, q.$$

4. *Matrix Transpose.* If $\overset{m\times n}{\mathbf{A}} = [a_{ij} : i = 1, \ldots, m; j = 1, \ldots, n]$, then the transpose of $\mathbf{A}$, written $\mathbf{A}^\top$, is given by the $n \times m$ matrix
$$\mathbf{A}^\top = [a_{ji} : j = 1, \ldots, n; i = 1, \ldots, m].$$
Note from the definitions that $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$.

5. *Matrix Inverse.* The inverse of a matrix $\overset{n\times n}{\mathbf{A}}$ (if it exists) is a matrix $\overset{n\times n}{\mathbf{B}}$ such that $\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$. We denote the inverse by $\mathbf{A}^{-1}$. Note that if $\mathbf{A}_1$ and $\mathbf{A}_2$ are both invertible, then $(\mathbf{A}_1\mathbf{A}_2)^{-1} = \mathbf{A}_2^{-1}\mathbf{A}_1^{-1}$.

6. *Trace.* The trace of a matrix $\overset{n\times n}{\mathbf{A}}$ is given by
$$\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}.$$

Lemma 2.1. *For any matrices* $\mathbf{A}$ *(*$n \times m$*) and* $\mathbf{B}$ *(*$m \times n$*),*
$$tr(\mathbf{AB}) = tr(\mathbf{BA}).$$

7. The *determinant* of a square matrix $\overset{n\times n}{\mathbf{A}}$ is defined as
$$\det(\mathbf{A}) = \sum (-1)^{|\tau|} a_{1\tau(1)} \ldots a_{n\tau(n)}$$
where the summation is taken over all permutations τ of $\{1, 2, \ldots, n\}$, and we define $|\tau| = 0$ or 1 depending on whether τ can be written as an even or odd number of transpositions.

E.g. If $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, then $\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21}$.

Proposition 2.1. *Matrix* $\overset{n\times n}{\mathbf{A}}$ *is invertible if and only if* $\det(\mathbf{A}) \neq 0$*. If* $\mathbf{A}^{-1}$ *exists then*
$$\det(\mathbf{A}) = \frac{1}{\det(\mathbf{A}^{-1})}$$

Proposition 2.2. *For any matrices* $\overset{n\times n}{\mathbf{A}}$, $\overset{n\times n}{\mathbf{B}}$, $\overset{n\times n}{\mathbf{C}}$ *such that* $\mathbf{C} = \mathbf{AB}$*,*
$$det(\mathbf{C}) = det(\mathbf{A}) \cdot det(\mathbf{B}).$$

2.1.3 Special matrices

Definition 2.1. An $n \times n$ matrix $\mathbf{A}$ is symmetric if

$$\mathbf{A} = \mathbf{A}^\top.$$

An $n \times n$ symmetric matrix $\mathbf{A}$ is **positive-definite** if

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

and is **positive semi-definite** if

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n.$$

$\mathbf{A}$ is **idempotent** if $\mathbf{A}^2 = \mathbf{A}$.

2.1.4 Vector Differentiation

Consider a real-valued function $f : \mathbb{R}^p \to \mathbb{R}$ of a vector variable $\mathbf{x} = (x_1, \ldots, x_p)^\top$. Sometimes we will want to differentiate f. We define the partial derivative of $f(\mathbf{x})$ with respect to $\mathbf{x}$ to be the vector of partial derivatives, i.e.

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ .. \\ .. \\ .. \\ \frac{\partial f}{\partial x_p}(\mathbf{x}) \end{bmatrix} \tag{2.1}$$

The following examples can be worked out directly from the definition (2.1), using the chain rule in some cases.

Example 2.1. If $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$ where $\mathbf{a} \in \mathbb{R}^p$ is a constant vector, then

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) = \mathbf{a}.$$

Example 2.2. If $f(\mathbf{x}) = (\mathbf{x} - \mathbf{a})^\top \mathbf{A}(\mathbf{x} - \mathbf{a})$ for a fixed vector $\mathbf{a} \in \mathbb{R}^p$ and $\mathbf{A}$ is a symmetric constant $p \times p$ matrix, then

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) = 2\mathbf{A}(\mathbf{x} - \mathbf{a}).$$

Example 2.3. Suppose that $g : \mathbb{R} \to \mathbb{R}$ is a differentiable function with derivative g'. Then, using the chain rule for partial derivatives,

$$\frac{\partial g(\mathbf{a}^\top \mathbf{x})}{\partial \mathbf{x}} = g'(\mathbf{a}^\top \mathbf{x}) \frac{\partial}{\partial \mathbf{x}} \left\{ \mathbf{a}^\top \mathbf{x} \right\} = g'(\mathbf{a}^\top \mathbf{x})\mathbf{a}.$$

Example 2.4. If f is defined as in Example 2.2 and g is as in Example 2.3 then, using the chain rule again,

$$\frac{\partial}{\partial \mathbf{x}} g\{f(\mathbf{x})\} = g'\{f(\mathbf{x})\}\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) = 2g'\{(\mathbf{x}-\mathbf{a})^\top \mathbf{A}(\mathbf{x}-\mathbf{a})\}\mathbf{A}(\mathbf{x}-\mathbf{a}).$$

If we wish to find a maximum or minimum of $f(\mathbf{x})$ we should search for stationary points of f, i.e. solutions to the system of equations

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ .. \\ .. \\ .. \\ \frac{\partial f}{\partial x_p}(\mathbf{x}) \end{bmatrix} = \mathbf{0}_p.$$

Definition 2.2. The **Hessian** matrix of f is the $p \times p$ matrix of second derivatives.

$$\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}^\top}(\mathbf{x}) = \left\{ \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_k} \right\}_{j,k=1}^{p}.$$

The nature of a stationary point is determined by the Hessian

If the Hessian is positive (negative) definite at a stationary point $\mathbf{x}$, then the stationary point is a minimum (maximum).

If the Hessian has both positive and negative eigenvalues at $\mathbf{x}$ then the stationary point will be a *saddle point*.

2.2 Vector spaces

It will be useful to talk about **vector spaces**. These are sets of vectors that can be added together, or multiplied by a scalar. You should be familiar with these from your undergraduate degree. We don't provide a formal definition here, but you can think of a real vector space V as a set of vectors such that for any $\mathbf{v}_1, \mathbf{v}_2 \in V$ and $\alpha_1, \alpha_2 \in \mathbb{R}$, we have

$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 \in V$$

i.e., vector spaces are closed under addition and scalar multiplication.

Example 2.5. Euclidean space in p dimensions, $\mathbb{R}^p$, is a vector space. If we add any two vectors in $\mathbb{R}^p$, or multiply a vector by a real scalar, then the resulting vector also lies in $\mathbb{R}^p$.

A subset $U \subset V$ of a vector space V is called a vector **subspace** if U is also a vector space.

Example 2.6. Let $V = \mathbb{R}^2$. Then the sets

$$U_1 = \left\{ \begin{pmatrix} a \\ 0 \end{pmatrix} : a \in \mathbb{R} \right\}, \text{ and } \quad U_2 = \left\{ a \begin{pmatrix} 1 \\ 1 \end{pmatrix} : a \in \mathbb{R} \right\}$$

are both subspaces of V.

2.2.1 Linear independence

Definition 2.3. Vectors $\overset{n\times 1}{\mathbf{x}}_1, \ldots, \overset{n\times 1}{\mathbf{x}}_p$ are said to be **linearly dependent** if there exist scalars $\lambda_1, \ldots, \lambda_p$ not all zero such that

$$\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \cdots + \lambda_p \mathbf{x}_p = \mathbf{0}.$$

Otherwise, these vectors are said to be **linearly independent**.

Definition 2.4. Given a set of vectors $S = \{s_1, \ldots, s_n\}$, the **span** of S is the smallest vector space containing S or equivalently, is the set of all linear combinations of vectors from S

$$\text{span}(S) = \left\{ \sum_{i=1}^{k} \alpha_i s_i \mid k \in \mathbb{N}, \alpha_i \in \mathbb{R}, s_i \in S \right\}$$

Definition 2.5. A **basis** of a vector space V is a set of linearly independent vectors in V that span V.

Example 2.7. Consider $V = \mathbb{R}^2$. Then the following are both bases for V:

$$B_1 = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

$$B_2 = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\}$$

Definition 2.6. The **dimension** of a vector space is the number of vectors in its basis.

2.2.2 Row and column spaces

We can think about the matrix-vector multiplication $\mathbf{Ax}$ in two ways. The usual way is as the inner product between the rows of A and x.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 + 2x_2 \\ 3x_1 + 4x_2 \\ 5x_1 + 6x_2 \end{pmatrix}$$

But a better way to think of $\mathbf{Ax}$ is as a linear combination of the columns of A.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix} + x_2 \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$$

Definition 2.7. The **column space** of a $n \times p$ matrix $\mathbf{A}$ is the set of all linear combinations of the columns of $\mathbf{A}$:

$$\mathcal{C}(\mathbf{A}) = \{\mathbf{Ax} : \mathbf{x} \in \mathbb{R}^p\} \subset \mathbb{R}^n$$

For

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

we can see that the column space is a 2-dimensional plane in $\mathbb{R}^3$. The matrix $\mathbf{B}$ has the same column space as $\mathbf{A}$

$$\mathbf{B} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 7 & 10 \\ 5 & 6 & 11 & 16 \end{pmatrix}$$

The number of linearly independent columns of $\mathbf{A}$ is called the **column rank** of $\mathbf{A}$, and is equal to the dimension of the column space of $\mathcal{C}(\mathbf{A})$. The **column rank** of $\mathbf{A}$ and $\mathbf{B}$ is 2.

The **row space** of $\mathbf{A}$ is defined to be the column space of $\mathbf{A}^\top$, and the **row rank** is the number of linearly independent rows of $\mathbf{A}$.

Theorem 2.1. *The **row rank** of a matrix equals the **column rank**.*

Thus we can simply refer to the **rank** of the matrix.

Proof. The proof of this theorem is very simple. Let $\mathbf{C}$ be an $n \times r$ matrix (where $r = \text{rank}(\mathbf{A})$) with columns chosen to be a set of r linearly independent columns from A. Then we know each column of $\mathbf{A}$ can be written as a linear combination of the columns of $\mathbf{C}$, i.e.

$$\mathbf{A} = \mathbf{CR}.$$

The dimension of $\mathbf{R}$ must be $r \times p$. But now we can see that the rows of $\mathbf{A}$ are formed by a linear combination of the rows of $\mathbf{R}$. Thus the row rank of $\mathbf{A}$ is at most r (=the column rank of $\mathbf{A}$). This holds for any matrix, so is true for $\mathbf{A}^\top$: namely row-rank$(A^\top) \leq$ column-rank$(A^\top)$. But the row space of $\mathbf{A}^\top$ equals $\mathcal{C}(\mathbf{A})$, thus proving the theorem! □

Corollary 2.1. *The rank of an $n \times p$ matrix is at most* $\min(n, p)$.

Example 2.8.

$$B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Example 2.9.

$$D = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

So the rank of D is 1.

2.2.3 Linear transformations

We can view an $n \times p$ matrix $\mathbf{A}$ as a linear map between two vector spaces:

$$\begin{aligned} \mathbf{A}: \ & \mathbb{R}^p \to \mathbb{R}^n \\ & \mathbf{x} \mapsto \mathbf{A}\mathbf{x} \end{aligned}$$

The **image** of $\mathbf{A}$ is precisely the column space of $\mathbf{A}$:

$$\mathrm{Im}(\mathbf{A}) = \{\mathbf{A}\mathbf{x} : \mathbf{x} \in \mathbb{R}^p\} = \mathcal{C}(\mathbf{A}) \subset \mathbb{R}^n$$

The **kernel** of A is the set of vectors mapped to zero:

$$\mathrm{Ker}(\mathbf{A}) = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{0}\} \subset \mathbb{R}^p$$

and is sometimes called the **null-space** of $\mathbf{A}$ and denoted $\mathcal{N}(\mathbf{A})$.

Theorem 2.2. *The **rank-nullity** theorem says if V and W are vector spaces, and $A : V \to W$ is a linear map, then*

$$\dim \mathrm{Im}(A) + \dim \mathrm{Ker}(A) = \dim V$$

If we're thinking about matrices, then $\dim \mathcal{C}(\mathbf{A}) + \dim \mathcal{N}(\mathbf{A}) = p$, or equivalently that
$\mathrm{rank}(\mathbf{A}) + \dim \mathcal{N}(\mathbf{A}) = p$.

We've already said that the row space of $\mathbf{A}$ is $\mathcal{C}(\mathbf{A}^\top)$. The left-null space is $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^\top \mathbf{A} = 0\}$ or equivalently $\{x \in \mathbb{R}^n : \mathbf{A}^\top \mathbf{x} = 0\} = \mathcal{N}(\mathbf{A}^\top)$. And so by the rank-nullity theorem we must have

$$n = \dim \mathcal{C}(\mathbf{A}^\top) + \dim \mathcal{N}(\mathbf{A}^\top) = \mathrm{rank}(\mathbf{A}) + \dim \mathrm{Ker}(\mathbf{A}^\top).$$

Example 2.10. Consider again the matrix $D : \mathbb{R}^3 \to \mathbb{R}^2$

$$D = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

We have already seen that

$$\mathcal{C}(D) = \text{span}\left\{\begin{pmatrix} 1 \\ 2 \end{pmatrix}\right\}$$

and so $\dim \mathcal{C}(D) = \text{rank}(D) = 1$. The kernel, or null-space, of $\mathbf{D}$ is the set of vectors for which $\mathbf{Dx} = \mathbf{0}$, i.e.,

$$x_1 + 2x_2 + 3x_3 = 0$$

This is a single equation with three unknowns, and so there must be a plane of solutions. We need two linearly independent vectors in this plane to describe it. Convince yourself that

$$\mathcal{N}(D) = \text{span}\left\{\begin{pmatrix} 0 \\ 3 \\ -2 \end{pmatrix}, \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}\right\}$$

So we have

$$\dim \mathcal{C}(D) + \dim \mathcal{N}(D) = 1 + 2 = 3$$

as required by the rank-nullity theorem.

If we consider $D^\top$, we already know $\dim \mathcal{C}(D^\top) = 1$ (as row-rank=column rank), and the rank-nullity theorem tells us that the dimension of the null space of $D^\top$ must be $2 - 1 = 1$. This is easy to confirm as $D^\top x = 0$ implies

$$x_1 + 2x_2 = 0$$

which is a line in $\mathbb{R}^2$

$$\mathcal{N}(D^\top) = \text{span}\left\{\begin{pmatrix} -2 \\ 1 \end{pmatrix}\right\}$$

Question: When does a square matrix $\mathbf{A}$ have an inverse?

- Precisely when the kernel of $\mathbf{A}$ contains only the zero vector, i.e., has dimension 0. In this case the column space of $\mathbf{A}$ is the original space, and $\mathbf{A}$ is surjective and so must have an inverse. A simpler way to determine if $\mathbf{A}$ has an inverse is to consider its determinant.

Question: Suppose we are given a $n \times p$ matrix $\mathbf{A}$, and a n-vector $\mathbf{y}$. When does

$$\mathbf{Ax} = \mathbf{y}$$

have a solution?

- When $\mathbf{y}$ is in the column space of $\mathbf{A}$,

$$\mathbf{y} \in \mathcal{C}(\mathbf{A})$$

Question: When is the answer unique?

- Suppose $\mathbf{x}$ and $\mathbf{x}'$ are both solutions with $\mathbf{x} \neq \mathbf{x}'$. We can write $\mathbf{x}' = \mathbf{x} + \mathbf{u}$ for some vector $\mathbf{u}$ and note that

$$\mathbf{y} = \mathbf{A}\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{u} = \mathbf{y} + \mathbf{A}\mathbf{u}$$

  and so $\mathbf{A}\mathbf{u} = \mathbf{0}$, i.e., $\mathbf{u} \in \mathcal{N}(A)$. So there are multiple solutions when the null-space of $\mathbf{A}$ contains more than the zero vector. If the dimension of $\mathcal{N}(A)$ is one, there is a line of solutions. If the dimension is two, there is a plane of solutions, etc.

2.3 Inner product spaces

2.3.1 Distances, and angles

Vector spaces are not particularly interesting from a statistical point of view until we equip them with a sense of geometry, i.e. distance and angle.

Definition 2.8. A real **inner product space** $(V, \langle \cdot, \cdot \rangle)$ is a real vector space V equipped with a map

$$\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$$

such that

1. $\langle \cdot, \cdot \rangle$ is a linear map in both arguments:

$$\langle \alpha \mathbf{v}_1 + \beta \mathbf{v}_2, \mathbf{u} \rangle = \alpha \langle \mathbf{v}_1, \mathbf{u} \rangle + \beta \langle \mathbf{v}_2, \mathbf{u} \rangle$$

for all $\mathbf{v}_1, \mathbf{v}_2, \mathbf{u} \in V$ and $\alpha, \beta \in \mathbb{R}$. 2. $\langle \cdot, \cdot \rangle$ is symmetric in its arguments: $\langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$ for all $\mathbf{u}, \mathbf{v} \in V$ 3. $\langle \cdot, \cdot \rangle$ is positive definite: $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$ for all $\mathbf{v} \in V$ with equality if and only if $\mathbf{v} = \mathbf{0}$.

An inner product provides a vector space with the concepts of

- **distance**: for all $v \in V$ define the **norm** of v to be

$$||\mathbf{v}|| = \langle \mathbf{v}, \mathbf{v} \rangle^{\frac{1}{2}}$$

  Thus any inner-product space $(V, \langle \cdot, \cdot \rangle)$ is also a normed space $(V, || \cdot ||)$, and a metric space $(V, d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||)$.

- **angle**: for $\mathbf{u}, \mathbf{v} \in V$ we define the angle between $\mathbf{u}$ and $\mathbf{v}$ to be θ where

$$\langle \mathbf{u}, \mathbf{v} \rangle = ||\mathbf{u}||.||\mathbf{v}|| \cos\theta$$
$$\implies \theta = \cos^{-1}\left(\frac{\langle \mathbf{u}, \mathbf{v} \rangle}{||\mathbf{u}||\,||\mathbf{v}||}\right)$$

  We will primarily be interested in the concept of **orthogonality**. We say $\mathbf{u}, \mathbf{v} \in V$ are orthogonal if

$$\langle \mathbf{u}, \mathbf{v} \rangle = 0$$

  i.e., the *angle* between them is $\frac{\pi}{2}$.

If you have done any functional analysis, you may recall that a Hilbert space is a *complete* inner-product space, and a Banach space is a complete normed space. This is an applied module, so we will skirt much of the technical detail, but note that some of the proofs formally require us to be working in a Banach or Hilbert space. We will not concern ourselves with such detail.

Example 2.11. We will mostly be working with the Euclidean vector spaces $V = \mathbb{R}^n$, in which we use the *Euclidean* inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\top \mathbf{v}$$

sometimes called the **scalar** or **dot product** of $\mathbf{u}$ and $\mathbf{v}$. Sometimes this gets weighted by a matrix so that

$$\langle \mathbf{u}, \mathbf{v} \rangle_Q = \mathbf{u}^\top \mathbf{Q} \mathbf{v}.$$

The norm associated with the dot product is the square root of the sum of squared errors, denoted by $||\cdot||_2$. The **length** of $\mathbf{u}$ is then

$$||\mathbf{u}||_2 = \sqrt{\mathbf{u}^\top \mathbf{u}} = \left(\sum_{i=1}^{n} u_i^2 \right)^{\frac{1}{2}} \geq 0.$$

Note that $||\mathbf{u}||_2 = 0$ if and only if $\mathbf{u} = \mathbf{0}_n$ where $\overset{n\times 1}{\mathbf{0}}_n = (0, 0, \ldots, 0)^\top$.

We say $\mathbf{u}$ is orthogonal to $\mathbf{v}$ if $\mathbf{u}^\top \mathbf{v} = 0$. For example, if

$$\mathbf{u} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ and } \mathbf{v} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$$

then

$$||\mathbf{u}||_2 = \sqrt{5} \text{ and } \mathbf{u}^\top \mathbf{v} = 0.$$

We will write $\mathbf{u} \perp \mathbf{v}$ if $\mathbf{u}$ is orthogonal to $\mathbf{v}$.

Definition 2.9. p-norm: The subscript 2 hints at a wider family of norms. We define the L_p norm to be

$$||\mathbf{v}||_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}.$$

2.3.2 Orthogonal matrices

Definition 2.10. A **unit vector** $\mathbf{v}$ is a vector satisfying $||\mathbf{v}|| = 1$, i.e., it is a vector of length 1. Vectors $\mathbf{u}$ and $\mathbf{v}$ are orthonormal if

$$||\mathbf{u}|| = ||\mathbf{v}|| = 1 \text{ and } \langle \mathbf{u}, \mathbf{v} \rangle = 0.$$

An $n \times n$ matrix $\mathbf{Q}$ is an **orthogonal matrix** if

$$\mathbf{Q}\mathbf{Q}^\top = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_n.$$

Equivalently, a matrix $\mathbf{Q}$ is orthogonal if $\mathbf{Q}^{-1} = \mathbf{Q}^\top$.

If $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ is an orthogonal matrix, then the columns $\mathbf{q}_1, \ldots, \mathbf{q}_n$ are mutually **orthonormal** vectors, i.e.

$$\mathbf{q}_j^\top \mathbf{q}_k = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k. \end{cases}$$

Lemma 2.2. *Let $\mathbf{Q}$ be a $n \times p$ matrix and suppose $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_p$, where $\mathbf{I}_p$ is the $p \times p$ identity matrix. If $\mathbf{Q}$ is a square matrix ($n = p$), then $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}_p$. If $\mathbf{Q}$ is not square ($n \neq p$), then $\mathbf{Q}\mathbf{Q}^\top \neq I_n$.*

Proof. Suppose $n = p$, and think of $\mathbf{Q}$ as a linear map

$$\begin{aligned} \mathbf{Q} :& \mathbb{R}^n \to \mathbb{R}^n \\ & \mathbf{v} \mapsto \mathbf{Q}\mathbf{v} \end{aligned}$$

By the rank-nullity theorem,

$$\dim \operatorname{Ker}(\mathbf{Q}) + \dim \operatorname{Im}(\mathbf{Q}) = n$$

and because $\mathbf{Q}$ has a left-inverse, we must have $\dim \operatorname{Ker}(\mathbf{Q}) = 0$, as otherwise $\mathbf{Q}^\top$ would have to map from a vector space of dimension less than n to $\mathbb{R}^n$. So $\mathbf{Q}$ is of full rank, and thus must also have a right inverse, $\mathbf{B}$ say, with $\mathbf{Q}\mathbf{B} = \mathbf{I}_n$. If we left multiply by $\mathbf{Q}^\top$ we get

$$\begin{aligned} \mathbf{Q}\mathbf{B} &= \mathbf{I}_n \\ \mathbf{Q}^\top\mathbf{Q}\mathbf{B} &= \mathbf{Q}^\top \\ \mathbf{I}_n\mathbf{B} &= \mathbf{Q}^\top \\ \mathbf{B} &= \mathbf{Q}^\top \end{aligned}$$

and so we have that $\mathbf{Q}^{-1} = \mathbf{Q}^\top$.

Now suppose $\mathbf{Q}$ is $n \times p$ with $n \neq p$. Then as $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_{p\times p}$, we must have $\operatorname{tr}(\mathbf{Q}^\top\mathbf{Q}) = p$. This implies that

$$\operatorname{tr}(\mathbf{Q}\mathbf{Q}^\top) = \operatorname{tr}(\mathbf{Q}^\top\mathbf{Q}) = m$$

and so we cannot have $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}_n$ as $\operatorname{tr} \mathbf{I}_n = n$. □

Corollary 2.2. *If $\mathbf{q}_1, \ldots, \mathbf{q}_n$ are mutually orthogonal $n \times 1$ unit vectors then*

$$\sum_{i=1}^{n} \mathbf{q}_i \mathbf{q}_i^\top = \mathbf{I}_n.$$

Proof. Let $\mathbf{Q}$ be the matrix with i^{th} column $\mathbf{q}_i$

$$\mathbf{Q} = \begin{pmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ | & & | \end{pmatrix}.$$

Then $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_n$, and $\mathbf{Q}$ is $n \times n$. Thus by Lemma 2.2, we must also have $\mathbf{Q}\mathbf{Q}^\top = \mathbf{I}_n$ and if we think about matrix-matrix multiplication as columns times rows (c.f. section 3.1), we get

$$\mathbf{I}_n = \mathbf{Q}\mathbf{Q}^\top = \begin{pmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ | & & | \end{pmatrix} \begin{pmatrix} - & \mathbf{q}_1^\top & - \\ & \vdots & \\ - & \mathbf{q}_n^\top & - \end{pmatrix} = \sum_{i=1}^{n} \mathbf{q}_i \mathbf{q}_i^\top$$

as required.

□

2.3.3 Projections

Definition 2.11. $\overset{n\times n}{\mathbf{P}}$ is a *projection* matrix if

$$\mathbf{P}^2 = \mathbf{P}$$

i.e., if it is idempotent.

View $\mathbf{P}$ as a map from a vector space W to itself. Let $U = \mathrm{Im}(\mathbf{P})$ and $V = \mathrm{Ker}(\mathbf{P})$ be the image and kernel of $\mathbf{P}$.

Proposition 2.3. *We can write $\mathbf{w} \in W$ as the sum of $\mathbf{u} \in U$ and $\mathbf{v} \in V$.*

Proof. Let $\mathbf{w} \in W$. Then

$$\mathbf{w} = \mathbf{I}_n \mathbf{w} = (\mathbf{I} - \mathbf{P})\mathbf{w} + \mathbf{P}\mathbf{w}$$

Now $\mathbf{P}\mathbf{w} \in \mathrm{Im}(\mathbf{P})$ and $(\mathbf{I} - \mathbf{P})\mathbf{w} \in \mathrm{Ker}(\mathbf{P})$ as

$$\mathbf{P}(\mathbf{I} - \mathbf{P})\mathbf{w} = (\mathbf{P} - \mathbf{P}^2)\mathbf{w} = \mathbf{0}.$$

□

Proposition 2.4. *If $\overset{n\times n}{\mathbf{P}}$ is a projection matrix then $\mathbf{I}_n - \mathbf{P}$ is also a projection matrix.*

The kernel and image of $\mathbf{I} - \mathbf{P}$ are the image and kernel (respectively) of $\mathbf{P}$:

$$\mathrm{Ker}(\mathbf{I} - \mathbf{P}) = U = \mathrm{Im}(\mathbf{P})$$
$$\mathrm{Im}(\mathbf{I} - \mathbf{P}) = V = \mathrm{Ker}(\mathbf{P}).$$

2.3.3.1 Orthogonal projection

We are mostly interested in **orthogonal** projections.

Definition 2.12. If W is an inner product space, and U is a subspace of W, then the orthogonal projection of $\mathbf{w} \in W$ onto U is the unique element $\mathbf{u} \in U$ that minimizes

$$||\mathbf{w} - \mathbf{u}||.$$

In other words, the orthogonal projection of $\mathbf{w}$ onto U is the *best possible approximation* of $\mathbf{w}$ in U.

As above, we can split W into U and its orthogonal compliment

$$U^{\perp} = \{\mathbf{x} \in W : \langle \mathbf{x}, \mathbf{u} \rangle = 0\}$$

i.e., $W = U \oplus U^{\perp}$ so that any $\mathbf{w} \in W$ can be written as $\mathbf{w} = \mathbf{u} + \mathbf{v}$ with $\mathbf{u} \in U$ and $\mathbf{v} \in U^{\perp}$.

Proposition 2.5. *If $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ is a basis for U, then the orthogonal projection matrix (i.e., the matrix that projects $\mathbf{w} \in W$ onto U) is*

$$\mathbf{P}_U = \mathbf{A}(\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}$$

where $\mathbf{A} = [\mathbf{u}_1 \ \ldots \ \mathbf{u}_k]$ is the matrix with columns given by the basis vectors.

Proof. We need to find $\mathbf{u} = \sum \lambda_i \mathbf{u}_i = \mathbf{A}\boldsymbol{\lambda}$ that minimizes $||\mathbf{w} - \mathbf{u}||$.

$$\begin{aligned} ||\mathbf{w} - \mathbf{u}||^2 &= \langle \mathbf{w} - \mathbf{u}, \mathbf{w} - \mathbf{u} \rangle \\ &= \mathbf{w}^{\top}\mathbf{w} - 2\mathbf{u}^{\top}\mathbf{w} + \mathbf{u}^{\top}\mathbf{u} \\ &= \mathbf{w}^{\top}\mathbf{w} - 2\boldsymbol{\lambda}^{\top}\mathbf{A}^{\top}\mathbf{w} + \boldsymbol{\lambda}^{\top}\mathbf{A}^{\top}\mathbf{A}\boldsymbol{\lambda}. \end{aligned}$$

Differentiating with respect to $\boldsymbol{\lambda}$ and setting equal to zero gives

$$\mathbf{0} = -2\mathbf{A}^{\top}\mathbf{w} + 2\mathbf{A}^{\top}\mathbf{A}\boldsymbol{\lambda}$$

and hence

$$\boldsymbol{\lambda} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{w}.$$

The orthogonal projection of $\mathbf{w}$ is hence

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{A}(\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{w}$$

and the projection matrix is

$$\mathbf{P}_U = \mathbf{A}(\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}.$$

□

Notes:

1. If $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ is an orthonormal basis for U then $\mathbf{A}^\top\mathbf{A} = \mathbf{I}$ and $\mathbf{P}_U = \mathbf{A}\mathbf{A}^\top$. We can then write

$$\mathbf{P}_U\mathbf{w} = \sum_i (\mathbf{u}_i^\top \mathbf{w})\mathbf{u}_i$$

and

$$\mathbf{P}_U = \sum_{i=1}^{k} \mathbf{u}_i\mathbf{u}_i^\top.$$

Note that if $U = W$ (so that $\mathbf{P}_U$ is a projection from W onto W, i.e., the identity), then $\mathbf{A}$ is a square matrix ($n \times n$) and thus $\mathbf{A}^\top\mathbf{A} = \mathbf{I}_n \implies \mathbf{A}\mathbf{A}^\top$ and thus $\mathbf{P}_U = \mathbf{I}_n$ as required. The coordinates (with respect to the orthonormal basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$) of a point $\mathbf{w}$ projected onto U are $\mathbf{A}^\top\mathbf{w}$.

2. $\mathbf{P}_U^2 = \mathbf{P}_U$, so $\mathbf{P}_U$ is a projection matrix in the sense of definition 2.11.

3. $\mathbf{P}_U$ is symmetric ($\mathbf{P}_U^\top = \mathbf{P}_U$). This is true for orthogonal projection matrices, but not in general for projection matrices.

Example 2.12. Consider the vector space $\mathbb{R}^2$ and let $\mathbf{u} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. The projection of $\mathbf{v} \in \mathbb{R}^2$ onto $\mathbf{u}$ is given by $(\mathbf{v}^\top\mathbf{u})\mathbf{u}$. So for example, if $\mathbf{v} = (2,\ 1)^\top$, then its projection onto $\mathbf{u}$ is

$$\mathbf{P}_U\mathbf{v} = \frac{3}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Alternatively, if we treat $\mathbf{u}$ as a basis for U, then the coordinate of $\mathbf{P}_U\mathbf{v}$ with respect to the basis is 3. To check this, draw a picture!

2.3.3.2 Geometric interpretation of linear regresssion

Consider the linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the vector of observations, $\mathbf{X}$ is the $n \times p$ design matrix, $\boldsymbol{\beta}$ is the $p \times 1$ vector of parameters that we wish to estimate, and $\mathbf{e}$ is a $n \times 1$ vector of zero-mean errors.

Least-squares regression tries to find the value of $\boldsymbol{\beta} \in \mathbb{R}^p$ that minimizes the sum of squared errors, i.e., we try to find $\boldsymbol{\beta}$ to minimize

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$$

We know that $\mathbf{X}\boldsymbol{\beta}$ is in the column space of $\mathbf{X}$, and so we can see that linear regression aims to find the *orthogonal projection* onto $\mathcal{C}(X)$.

$$\mathbf{P}_U\mathbf{y} = \arg \min_{\mathbf{y}' : \mathbf{y}' \in \mathcal{C}(X)} ||\mathbf{y} - \mathbf{y}'||_2.$$

By Proposition 2.5 this is

$$\mathbf{P}_U\mathbf{y} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \hat{\mathbf{y}}$$

which equals the usual prediction obtained in linear regression ($\hat{\mathbf{y}}$ are often called the fitted values). We can also see that the choice of $\boldsymbol{\beta}$ that specifies this point in $\mathcal{C}(X)$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

which is the usual least-squares estimator.

2.4 The Centering Matrix

The **centering matrix** will be play an important role in this module, as we will use it to remove the column means from a matrix (so that each column has mean zero), *centering* the matrix.

Definition 2.13. The **centering matrix** is

$$\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top. \tag{2.2}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix, and $\mathbf{1}_n$ is an $n \times 1$ column vector of ones.

You will be asked to prove the following results about $\mathbf{H}$ in the exercises:

1. The matrix $\mathbf{H}$ is a projection matrix, i.e. $\mathbf{H}^\top = \mathbf{H}$ and $\mathbf{H}^2 = \mathbf{H}$.
2. Writing $\mathbf{0}_n$ for the $n \times 1$ vector of zeros, we have $\mathbf{H}\mathbf{1}_n = \mathbf{0}_n$ and $\mathbf{1}_n^\top\mathbf{H} = \mathbf{0}_n^\top$. In words: the sum of each row and each column of $\mathbf{H}$ is 0.
3. If $\mathbf{x} = (x_1, \ldots, x_n)^\top$, then $\mathbf{H}\mathbf{x} = \mathbf{x} - \bar{x}\mathbf{1}_n$ where $\bar{x} = n^{-1}\sum_{i=1}^n x_i$. I.e., H subtracts the mean $\bar{x}$ from $\mathbf{x}$.
4. With $\mathbf{x}$ as in 3., we have

   $$\mathbf{x}^\top\mathbf{H}\mathbf{x} = \sum_{i=1}^n (x_i - \bar{x})^2,$$

   and so

   $$\frac{1}{n}\mathbf{x}^\top\mathbf{H}\mathbf{x} = \frac{1}{n}\sum_{i=1}^n (x_i - \bar{x})^2 = \hat{\sigma}^2,$$

   where $\hat{\sigma}^2$ is the sample variance.
5. If

   $$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1^\top & - \\ & \vdots & \\ - & \mathbf{x}_n^\top & - \end{bmatrix} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$$

   is an $n \times p$ data matrix containing data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$, then

   $$\mathbf{H}\mathbf{X} = \begin{bmatrix} - & (\mathbf{x}_1 - \bar{\mathbf{x}})^\top & - \\ - & (\mathbf{x}_2 - \bar{\mathbf{x}})^\top & - \\ & \vdots & \\ - & (\mathbf{x}_n - \bar{\mathbf{x}})^\top & - \end{bmatrix} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \ldots, \mathbf{x}_n - \bar{\mathbf{x}}]^\top$$

where

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \in \mathbb{R}^p$$

is the p-dimensional sample mean of $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$. In words, $\mathbf{H}$ has subtracted the column mean from each column of $\mathbf{X}$.

6. With $\mathbf{X}$ as in 5.

$$\frac{1}{n}\mathbf{X}^\top\mathbf{H}\mathbf{X} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top = \mathbf{S},$$

where $\mathbf{S}$ is the sample covariance matrix.

7. If $\mathbf{A} = (a_{ij})_{i,j=1}^n$ is a symmetric $n \times n$ matrix, then

$$\mathbf{B} = \mathbf{H}\mathbf{A}\mathbf{H} = \mathbf{A} - \mathbf{1}_n\bar{\mathbf{a}}_+^\top - \bar{\mathbf{a}}_+\mathbf{1}_n^\top + \bar{a}_{++}\mathbf{1}_n\mathbf{1}_n^\top,$$

or, equivalently,

$$b_{ij} = a_{ij} - \bar{a}_{i+} - \bar{a}_{+j} + \bar{a}_{++}, \qquad i, j = 1, \ldots, n,$$

where

$$\bar{\mathbf{a}}_+ \equiv (\bar{a}_{1+}, \ldots, \bar{a}_{n+})^\top = \frac{1}{n}\mathbf{A}\mathbf{1}_n,$$

$\bar{a}_{+j} = \bar{a}_{j+}$, for , $j = 1, \ldots, n$,, and , $\bar{a}_{++} = n^{-2}\sum_{i,j=1}^n a_{ij}$.

Note that Property 3. is a special case of Property 5., and Property 4. is a special case of Property 6. However, it is useful to see these results in the simpler scalar case before moving onto the general matrix case.

2.5 Computer tasks

This Chapter's computer tasks are short and sweet, as the focus has primarily been on the mathematics. Tasks for later chapters will be more challenging.

0. Let's consider some basic matrix computations in R. First, we show how to do matrix multiplication and addition

```
a=c(3,1,1,6)                      # define a column  vector a
b=c(5,6,2,8)                      # define a  vector b
A=matrix(a,nrow=2,byrow=TRUE)     # use a to define a matrix A
# Note that by default R fills a matrix by column. You have to explictly
# ask for it to be filled by row.
A
```

```
##      [,1] [,2]
## [1,]    3    1
## [2,]    1    6
```

```
B=matrix(b,nrow=2,byrow=TRUE)     # use b to define a matrix B
B
```

```
##      [,1] [,2]
## [1,]    5    6
## [2,]    2    8
```

```
A%*%B                             # use %*% to multiply two matrices
```

```
##      [,1] [,2]
## [1,]   17   26
## [2,]   17   54
```

```
                                  # together in the usual sense
A+B                               # add
```

```
##      [,1] [,2]
## [1,]    8    7
## [2,]    3   14
```

```
dim(A)                            # prints the dimension of a matrix.
```

```
## [1] 2 2
```

Multiplication of a matrix by a scalar is easy - but be careful if you use the * for two square matrices, as R will do element-wise multiplication

```
3*A
```

```
##      [,1] [,2]
## [1,]    9    3
## [2,]    3   18
```

```
A*B # compare with A%*%B
```

```
##      [,1] [,2]
## [1,]   15    6
## [2,]    2   48
```

Note that R won't let you multiply matrices that are not conformable (i.e. not the right shape).

The usual Euclidean inner product is just matrix multiplication

```
t(a) %*% b  # t() transposes a matrix
```

```
##      [,1]
## [1,]   71
```

The inverse, determinant, and trace of a matrix are computed as follows:

```
solve(A) # the inverse
```

```
##              [,1]        [,2]
## [1,]  0.35294118 -0.05882353
## [2,] -0.05882353  0.17647059
```

```
det(A)
```

```
## [1] 17
```

```
sum(diag(A)) # the trace is the sum of the diagonal elements of a matrix.
```

```
## [1] 9
```

Note that numerical errors will start to appear quite quickly. For example, the following should return the identity matrix. The result is very close to the identity, but not exactly equal to it. With larger matrices, numerical errors can be worse and appear alarmingly quickly.

```
A%*%solve(A)
```

```
##               [,1] [,2]
## [1,] 1.000000e+00    0
## [2,] 5.551115e-17    1
```

1. Solve the linear system for $\mathbf{x}$ using R.

$$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 2 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

2. Consider the `iris` dataset. Let $\mathbf{X}$ be the 4 numerical variables

```
X = as.matrix(iris[,1:4])
```

- Compute the sample mean vector, the sample covariance matrix, and the sample correlation matrix for the four numerical variables using the in built R commands `colMeans`, 'cov, and `cor`.
- Compute the centering matrix for $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$ using $n = 150$ (the number of data points in the iris dataset), and check compute the column means of $\mathbf{HX}$ are all zero (or close - there will be numerical error). Compute the sample covariance and correlation matrices using $\mathbf{H}$.

```
n=150
H=diag(rep(1,n))-rep(1,n)%*%t(rep(1,n))/n    # calculate the centering matrix H
```

- Check the properties of the centering matrix (you can ignore 7.) given in Section 2.4
- What does the following command do?

```
sweep(X, 2, colMeans(X))
```

Thus you'll see that it usually isn't worth computing the centering matrix when doing things in practice. We use $\mathbf{H}$ in the description of the methods as it makes the mathematics easier to write down.

- Compute the covariance matrix of $\mathbf{X}$ directly (ie, don't use the `cov` command - but do check your answer with `cov`).

2.6 Exercises

1. Are the vectors $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and $\begin{pmatrix} -1 \\ 2 \end{pmatrix}$ linearly independent?
   - Give two different bases for $\mathbb{R}^2$
   - Describe three different subspaces of $\mathbb{R}^3$
2. Let
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ 1 & 1 & 1 \end{pmatrix}$$
   - What is rank($\mathbf{A}$)?
   - Write the product $\mathbf{Ax}$ where $\mathbf{x}^\top = (x_1, x_2, x_3)$ as both the inner product of the rows of $\mathbf{A}$ and $\mathbf{x}$, and as a linear combination of the columns of $\mathbf{A}$ (see section 2.2.2)
   - Describe the column space of $\mathbf{A}$. What is its dimension?
   - Find a vector in the kernel of $\mathbf{A}$.
   - Describe the kernel of $\mathbf{A}$ as a vector space and give a basis for the space.
   - Is $\mathbf{A}$ invertible? What is det($\mathbf{A}$)?
3. Let's consider the inner product space $\mathbb{R}^3$ with the Euclidean inner product
$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$$
   Let
$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} -1 \\ 0 \\ -2 \end{pmatrix}. \quad \mathbf{x}_3 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$
   - What is the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$? Which pairs of vectors are orthogonal to each other?
   - What is the norm associated with this inner-product space, and compute the norm of $\mathbf{x}_1, \ldots, \mathbf{x}_3$. What is the geometric interpretation of the norm?
4. Prove the following statements:
   - The determinant of an orthogonal matrix must be either 1 or -1.
   - If $\mathbf{A}$ and $\mathbf{B}$ are orthogonal matrices, then $\mathbf{AB}$ must also be orthogonal.

- Let $\mathbf{A}$ be an $n \times n$ matrix of the form

$$\mathbf{A} = \mathbf{Q}\mathbf{B}\mathbf{Q}^\top.$$

where $\mathbf{Q}$ is an $n \times n$ matrix, and $\mathbf{B}$ is an $n \times n$ diagonal matrix. Prove that $\mathbf{A}$ is symmetric.

5. Consider the matrix

$$\mathbf{P} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

- Show that $\mathbf{P}$ is a projection matrix.
- Describe the subspace $\mathbf{P}$ projects onto.
- Describe the image and kernel of $\mathbf{P}$.
- Repeat the above questions using $\mathbf{I} - \mathbf{P}$ and check proposition 2.4.

6. Let $W = \mathbb{R}^3$ with the usual inner product. Consider the orthogonal projection from W onto the subspace U defined by

$$U = \text{span}\left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

- What is projection of the vector

$$\mathbf{v} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

onto U? Show that this vector does minimize $||\mathbf{v} - \mathbf{u}||$ for $\mathbf{u} \in U$.
- Write down the orthogonal projection matrix for the projection W onto U and check it is a projection matrix. Check your answer to the previous part of the question.

7. The centering matrix will be play an important role in this module, as we will use it to remove the column means from a matrix (so that each column has mean zero), *centering* the matrix.

Define

$$\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$$

to be the $n \times n$ centering matrix (see 2.4).
Let $\mathbf{x} = (x_1, \ldots, x_n)^\top$ denote a vector and let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ denote an $n \times p$ data matrix.

Define the scalar sample mean to be $\bar{x} = n^{-1}\sum_{i=1}^n x_i$ and the sample mean vector to be $\bar{\mathbf{x}} = n^{-1}\sum_{i=1}^n \mathbf{x}_i$.

i. Show by direct calculation that $\mathbf{H}$ is a projection matrix, i.e. $\mathbf{H}^\top = \mathbf{H}$ and $\mathbf{H}^2 = \mathbf{H}$.

ii. Show that

$$\mathbf{H}\mathbf{x} = \mathbf{x} - \bar{x}\mathbf{1}_n = (x_1 - \bar{x}, \ldots, x_n - \bar{x})^\top.$$

Hint: first show that $n^{-1}\mathbf{1}_n^\top \mathbf{x} = \bar{x}$.

iii. Show that

$$\mathbf{x}^\top \mathbf{H}\mathbf{x} = \sum_{i=1}^{n} (x_i - \bar{x})^2.$$

Hint: use the fact that $\mathbf{H}$ is a projection matrix and hence express $\mathbf{x}^\top \mathbf{H}\mathbf{x}$ as a scalar product of $\mathbf{H}\mathbf{x}$ with itself.

iv. Assuming $\mathbf{X}$ is an $n \times p$ matrix, show that

$$\mathbf{H}\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \ldots, \mathbf{x}_n - \bar{\mathbf{x}}]^\top.$$

Hint: first show that $n^{-1}\mathbf{1}_n^\top \mathbf{X} = \bar{\mathbf{x}}^\top$.

v. Using $\mathbf{S}$ to denote the sample covariance matrix, show that

$$\mathbf{X}^\top \mathbf{H}\mathbf{X} = \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top = n\mathbf{S}, \tag{2.3}$$

Hint: using the fact that $\mathbf{H}$ is a projection matrix, show that $\mathbf{X}^\top \mathbf{H}\mathbf{X} = (\mathbf{H}\mathbf{X})^\top (\mathbf{H}\mathbf{X})$.

Comment: Equation (2.3) provides a convenient way to calculate the sample covariance matrix directly in R, given the data matrix $\mathbf{X}$.

Chapter 3

Matrix decompositions

This chapter focusses on two ways to decompose a matrix into smaller parts. We can then think about which are the most important parts of the matrix, and that will be useful when we think about dimension reduction. The highlight of the chapter is the singular value decomposition (SVD), which is one of the most useful mathematical concepts from the past century, and is relied upon throughout statistics and machine learning. The SVD extends the idea of the eigen (or spectral) decomposition of symmetric square matrices to any matrix.

- Matrix-matrix products
- Eigenvalues and the spectral decomposition
- Introduction to the singular value decomposition
- SVD optimization results
- Low-rank approximation

3.1 Matrix-matrix products

Before we can introduce the SVD, we first need to recap some basic material on matrix multiplication and eigenvalues. We saw in section 2.2.2 that we can think about matrix-vector products in two ways: $\mathbf{Ax}$ is rows of $\mathbf{A}$ times $\mathbf{x}$; or as a linear combination of the columns of $\mathbf{A}$. We can similarly think about matrix-matrix products in two ways.

The usual way to think about the matrix product $\mathbf{AB}$ is as the rows of $\mathbf{A}$ times the columns of $\mathbf{B}$:

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ a_{21} & a_{22} & a_{23} \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & b_{12} & \cdot \\ \cdot & b_{22} & \cdot \\ \cdot & b_{32} & \cdot \end{bmatrix}$$

A better way (for this module) to think of $\mathbf{AB}$ is as the columns of $\mathbf{A}$ times the rows of $\mathbf{B}$. If we let $\mathbf{a}_i$ denote the columns of $\mathbf{A}$, and $\mathbf{b}_i^*$ the rows of $\mathbf{B}$ then

$$\begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} - & \mathbf{b}_1^* & - \\ - & \mathbf{b}_2^* & - \\ - & \mathbf{b}_3^* & - \end{bmatrix} = \sum_{i=1}^{3} \mathbf{a}_i \mathbf{b}_i^*$$

i.e., $\mathbf{AB}$ is a sum of the columns of $\mathbf{A}$ times the rows of $\mathbf{B}$.

Note that if $\mathbf{a}$ is a vector of length n and $\mathbf{b}$ is a vector of length p then $\mathbf{ab}^\top$ is an $n \times p$ matrix.

Example 3.1.

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} (2\ 3\ 1) = \begin{pmatrix} 2 & 3 & 1 \\ 4 & 6 & 2 \end{pmatrix}.$$

Note that $\mathbf{ab}^\top$ is a rank-1 matrix as its columns are all multiples of $\mathbf{a}$, or in other words, its column space is just multiples of $\mathbf{a}$.

$$\mathcal{C}(\mathbf{ab}^\top) = \{\lambda \mathbf{a} : \lambda \in \mathbb{R}\}.$$

We sometimes call $\mathbf{ab}^\top$ the **outer product** of $\mathbf{a}$ with $\mathbf{b}$.

By thinking of matrix-matrix multiplication in this way

$$\mathbf{AB} = \sum_{i=1}^{k} \mathbf{a}_i \mathbf{b}_i^*$$

(where k is the number of columns of $\mathbf{A}$ and the number of rows of $\mathbf{B}$) we can see that the product is a sum of rank-1 matrices. We can think of rank-1 matrices as the building blocks of matrices.

This chapter is about ways of decomposing matrices into their most important parts, and we will do this by thinking about the most important rank-1 building blocks.

Firstly though, we need a recap on eigenvectors.

3.2 Spectral/eigen decomposition

3.2.1 Eigenvalues and eigenvectors

Consider the $n \times n$ matrix $\mathbf{A}$. We say that vector $\mathbf{x} \in \mathbb{R}^n$ is an **eigenvector** corresponding to **eigenvalue** λ of $\mathbf{A}$ if

$$\mathbf{Ax} = \lambda \mathbf{x}.$$

To find the eigenvalues of a matrix, we note that if λ is an eigenvalue, then $(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = \mathbf{0}$, i.e., the kernel of $\mathbf{A} - \lambda\mathbf{I}_n$ has dimension at least 1, so $\mathbf{A} - \lambda\mathbf{I}_n$ is not invertible, and so we must have $\det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$.

Let $R(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}_n)$, which is an n^{th} order polynomial in λ. To find the eigenvalues of $\mathbf{A}$ we find the n roots $\lambda_1, \ldots, \lambda_n$ of $R(\lambda)$. We will always consider ordered eigenvalues so that $\lambda_1 \geq \cdots \geq \lambda_n$

Proposition 3.1. *If $\mathbf{A}$ is symmetric (i.e. $\mathbf{A}^\top = \mathbf{A}$) then the eigenvalues and eigenvectors of $\mathbf{A}$ are* real *(in $\mathbb{R}$).*

Proposition 3.2. *If $\mathbf{A}$ is an $n \times n$ symmetric matrix then its determinant is the product of its eigenvalues, i.e. $det(\mathbf{A}) = \lambda_1 \ldots \lambda_n$.*

Thus,

$$\mathbf{A} \text{ is invertible } \iff \det(\mathbf{A}) \neq 0 \iff \lambda_i \neq 0 \forall i \iff \mathbf{A} \text{ is of full rank}$$

3.2.2 Spectral decomposition

The key to much of dimension reduction is finding matrix decompositions. The first decomposition we will consider is the **spectral decomposition** (also called an **eigen-decomposition**).

Proposition 3.3. ***(Spectral decomposition).*** *Any $n \times n$ symmetric matrix $\mathbf{A}$ can be written as*

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top = \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^\top,$$

where $\mathbf{\Lambda} = diag\{\lambda_1, \ldots, \lambda_n\}$ is an $n \times n$ diagonal matrix consisting of the eigenvalues of $\mathbf{A}$ and $\mathbf{Q}$ is an orthogonal matrix ($\mathbf{Q}\mathbf{Q}^\top = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_n$) whose columns are unit eigenvectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$ of $\mathbf{A}$.

Because $\mathbf{\Lambda}$ is a diagonal matrix, we sometimes refer to the spectral decomposition as **diagonalizing** the matrix $\mathbf{A}$ as $\mathbf{Q}^\top\mathbf{A}\mathbf{Q} = \mathbf{\Lambda}$ is a diagonal matrix.

This will be useful at various points throughout the module. Note that it relies upon the fact that the eigenvectors of $\mathbf{A}$ can be chosen to be mutually orthogonal, and as there are n of them, they form an orthonormal basis for $\mathbb{R}^n$.

Corollary 3.1. *The rank of a symmetric matrix is equal to the number of non-zero eigenvalues (counting according to their multiplicities).*

Proof. If r is the number of non-zero eigenvalues of $\mathbf{A}$, then we have (after possibly reordering the λ_i)

$$\mathbf{A} = \sum_{i=1}^{r} \lambda_i \mathbf{q}_i \mathbf{q}_i^\top.$$

Each $\mathbf{q}_i\mathbf{q}_i^\top$ is a rank 1 matrix, with column space equal to the span of $\mathbf{q}_i$. As the $\mathbf{q}_i$ are orthogonal, the columns spaces $\mathcal{C}(\mathbf{q}_i\mathbf{q}_i^\top)$ are orthogonal, and their union is a vector space of dimension r. Hence the rank of $\mathbf{A}$ is r. □

Lemma 3.1. *Let* $\mathbf{A}$ *be an* $n \times n$ *symmetric matrix with (necessarily real) eigenvalues* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. *Then* $\mathbf{A}$ *is* positive definite *if and only if* $\lambda_n > 0$. *It is positive semi-definite if and only if* $\lambda_n \geq 0$.

Proof. If $\mathbf{A}$ is positive definite, and if $\mathbf{x}$ is a unit-eigenvector of $\mathbf{A}$ corresponding to λ_n, then

$$0 < \mathbf{x}^\top \mathbf{A}\mathbf{x} = \lambda_n \mathbf{x}^\top \mathbf{x} = \lambda_n.$$

Conversely, suppose $\mathbf{A}$ has positive eigenvalues. Because $\mathbf{A}$ is real and symmetric, we can write it as $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$. Now if $\mathbf{x}$ is a non-zero vector, then $\mathbf{y} = \mathbf{Q}^\top \mathbf{x} \neq \mathbf{0}$, (as $\mathbf{Q}^\top$ has inverse $\mathbf{Q}$ and hence $\dim \mathrm{Ker}(\mathbf{Q}) = 0$). Thus

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \mathbf{y}^\top \mathbf{\Lambda}\mathbf{y} = \sum_{i=1}^{n} \lambda_i y_i^2 > 0$$

and thus $\mathbf{A}$ is positive definite. □

Note: A covariance matrix $\mathbf{\Sigma}$ is always positive semi- definite (and thus always has non-negative eigenvalues). To see this, recall that if $\mathbf{x}$ is a random vector with $\mathbb{V}\mathrm{ar}(\mathbf{x}) = \mathbf{\Sigma}$, then for any constant vector $\mathbf{a}$, the random variable $\mathbf{a}^\top \mathbf{x}$ has variance $\mathbb{V}\mathrm{ar}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}^\top \mathbf{\Sigma}\mathbf{a}$. Because variances are positive, we must have

$$\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a} \geq 0 \; \forall \; \mathbf{a}.$$

Moreover, if $\mathbf{\Sigma}$ is positive definite (so that its eigenvalues are positive), then its determinant will be positive (so that $\mathbf{\Sigma}$ is **non-singular**) and we can find an inverse $\mathbf{\Sigma}^{-1}$ matrix, which is called the **precision** matrix.

Proposition 3.4. *The eigenvalues of a projection matrix* $\mathbf{P}$ *are all* 0 *or* 1.

3.2.3 Matrix square roots

From the spectral decomposition theorem, we can see that if $\mathbf{A}$ is a symmetric positive semi-definite matrix, then for any integer p

$$\mathbf{A}^p = \mathbf{Q}\mathbf{\Lambda}^p \mathbf{Q}^\top.$$

If in addition $\mathbf{A}$ is positive definite (rather than just semi-definite), then

$$\mathbf{A}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^\top$$

where $\mathbf{\Lambda}^{-1} = \mathrm{diag}\{\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}\}$.

The spectral decomposition also gives us a way to define a matrix square root. If we assume $\mathbf{A}$ is positive semi-definite, then its eigenvalues are non-negative, and the diagonal elements of $\mathbf{\Lambda}$ are all non-negative.

We then define $\mathbf{A}^{1/2}$, a matrix square root of $\mathbf{A}$, to be $\mathbf{A}^{1/2} = \mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^\top$ where $\mathbf{\Lambda}^{1/2} = \text{diag}\{\lambda_1^{1/2}, \ldots, \lambda_n^{1/2}\}$. This definition makes sense because

$$\begin{aligned}\mathbf{A}^{1/2}\mathbf{A}^{1/2} &= \mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^\top\mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^\top \\ &= \mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}\mathbf{Q}^\top \\ &= \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top \\ &= \mathbf{A},\end{aligned}$$

where $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_n$ and $\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2} = \mathbf{\Lambda}$. The matrix $\mathbf{A}^{1/2}$ is not the only matrix square root of $\mathbf{A}$, but it *is* the only symmetric, positive semi-definite square root of $\mathbf{A}$.

If $\mathbf{A}$ is positive definite (as opposed to just positive semi-definite), then all the λ_i are positive and so we can also define $\mathbf{A}^{-1/2} = \mathbf{Q}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^\top$ where $\mathbf{\Lambda}^{-1/2} = \text{diag}\{\lambda_1^{-1/2}, \ldots, \lambda_n^{-1/2}\}$. Note that

$$\mathbf{A}^{-1/2}\mathbf{A}^{-1/2} = \mathbf{Q}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^\top\mathbf{Q}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^\top = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^\top = \mathbf{A}^{-1},$$

so that, as defined above, $\mathbf{A}^{-1/2}$ is the matrix square root of $\mathbf{A}^{-1}$. Furthermore, similar calculations show that

$$\mathbf{A}^{1/2}\mathbf{A}^{-1/2} = \mathbf{A}^{-1/2}\mathbf{A}^{1/2} = \mathbf{I}_n,$$

so that $\mathbf{A}^{-1/2}$ is the matrix inverse of $\mathbf{A}^{1/2}$.

3.3 Singular Value Decomposition (SVD)

The spectral decomposition theorem (Proposition 3.3) gives a decomposition of any symmetric matrix. We now give a generalisation of this result which applies to *all* matrices.

If matrix $\mathbf{A}$ is not a square matrix, then it cannot have eigenvectors. Instead, it has **singular vectors** corresponding to **singular values**. Suppose is a $n \times p$ matrix. Then we say σ is a **singular value** with corresponding **left** and **right** singular vectors $\mathbf{u}$ and $\mathbf{v}$ (respectively) if

$$\mathbf{A}\mathbf{v} = \sigma\mathbf{u} \quad \text{and} \quad \mathbf{A}^\top\mathbf{u} = \sigma\mathbf{v}$$

If $\mathbf{A}$ if a symmetric matrix then $\mathbf{u} = \mathbf{v}$ is a eigenvector and σ is an eigenvalue.

The singular value decomposition (SVD) **diagonalizes A** into a product of a matrix of left singular vectors $\mathbf{U}$, a diagonal matrix of singular values $\mathbf{\Sigma}$, and a matrix of right singular vectors $\mathbf{V}$.

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top.$$

Proposition 3.5. ***(Singular value decomposition)***. *Let* $\mathbf{A}$ *be a* $n \times p$ *matrix of rank* r*, where* $1 \leq r \leq \min(n, p)$*. Then there exists a* $n \times r$ *matrix* $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$*, a* $p \times r$ *matrix* $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$*, and a* $r \times r$ *diagonal matrix* $\mathbf{\Sigma} = diag\{\sigma_1, \ldots, \sigma_r\}$ *such that*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top,$$

where $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_r = \mathbf{V}^\top\mathbf{V}$ *and the* $\sigma_1 \geq \ldots \geq \sigma_r > 0$*.*

Note that the $\mathbf{u}_i$ and the $\mathbf{v}_i$ are necessarily unit vectors, and that we have ordered the singular values from largest to smallest. The scalars $\sigma_1, \ldots, \sigma_r$ are called the **singular values** of $\mathbf{A}$, the columns of $\mathbf{U}$ are the **left singular vectors**, and the columns of $\mathbf{V}$ are the **right singular vectors**.

The form of the SVD given above is called the **compact singular value decomposition**. Sometimes we write it in a non-compact form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

where $\mathbf{U}$ is a $n \times n$ orthogonal matrix ($\mathbf{U}^\top\mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}_n$), $\mathbf{V}$ is a $p \times p$ orthogonal matrix ($\mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}_p$), and $\mathbf{\Sigma}$ is a $n \times p$ diagonal matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \ldots & & & 0 \\ 0 & \sigma_2 & 0 & \ldots & & \\ \vdots & & & & & \\ 0 & 0 & & \ldots & \sigma_r & \\ 0 & 0 & & \ldots & 0 & \ldots \\ \vdots & & & & & \\ 0 & 0 & & \ldots & & 0 \end{pmatrix}. \tag{3.1}$$

The columns of $\mathbf{U}$ and $\mathbf{V}$ form an orthonormal basis for $\mathbb{R}^n$ and $\mathbb{R}^p$ respectively. We can see that we recover the compact form of the SVD by only using the first r columns of $\mathbf{U}$ and $\mathbf{V}$, and truncating $\mathbf{\Sigma}$ to a $r \times r$ matrix with non-zero diagonal elements.

When $\mathbf{A}$ is symmetric, we take $\mathbf{U} = \mathbf{V}$, and the spectral decomposition theorem is recovered, and in this case (but not in general) the singular values of $\mathbf{A}$ are eigenvalues of $\mathbf{A}$.

Proof. $\mathbf{A}^\top\mathbf{A}$ is a $p \times p$ symmetric matrix, and so by the spectral decomposition theorem we can write it as

$$\mathbf{A}^\top\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$$

where $\mathbf{V}$ is a $p \times p$ orthogonal matrix containing the orthonormal eigenvectors of $\mathbf{A}^\top\mathbf{A}$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_r)$ is a diagonal matrix of eigenvalues with $\lambda_1 \geq \ldots \geq \lambda_r > 0$ (by Corollary 3.1).

For $i = 1, \ldots, r$, let $\sigma_i = \sqrt{\lambda_i}$ and let $\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{A}\mathbf{v}_i$. Then the vectors $\mathbf{u}_i$ are orthonormal:

$$\begin{aligned}
\mathbf{u}_i^\top \mathbf{u}_j &= \frac{1}{\sigma_i \sigma_j} \mathbf{v}_i^\top \mathbf{A}^\top \mathbf{A} \mathbf{v}_j \\
&= \frac{\sigma_j^2}{\sigma_i \sigma_j} \mathbf{v}_i^\top \mathbf{v}_j \quad \text{as } \mathbf{v}_j \text{ is an eigenvector of } \mathbf{A}^\top \mathbf{A} \\
&= \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad \text{as the } \mathbf{v}_i \text{ are orthonormal vectors.}
\end{aligned}$$

In addition

$$\mathbf{A}^\top \mathbf{u}_i = \frac{1}{\sigma_i} \mathbf{A}^\top \mathbf{A} \mathbf{v}_i = \frac{\sigma_i^2}{\sigma_i} \mathbf{v}_i = \sigma_i \mathbf{v}_i$$

and so $\mathbf{u}_i$ and $\mathbf{v}_i$ are left and right singular vectors.

Let $\mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_2 \ \ldots \ \mathbf{u}_r \ \ldots \ \mathbf{u}_n]$, where $\mathbf{u}_{r+1}, \ldots, \mathbf{u}_n$ are chosen to complete the orthonormal basis for $\mathbb{R}^n$ given $\mathbf{u}_1, \ldots, \mathbf{u}_r$, and let Σ be the $n \times p$ diagonal matrix in Equation (3.1).

Then we have shown that

$$\mathbf{U} = \mathbf{A}\mathbf{V}\mathbf{\Sigma}^{-1}$$

Thus

$$\begin{aligned}
\mathbf{U} &= \mathbf{A}\mathbf{V}\mathbf{\Sigma}^{-1} \\
\mathbf{U}\mathbf{\Sigma} &= \mathbf{A}\mathbf{V} \\
\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top &= \mathbf{A}.
\end{aligned}$$

□

Note that by construction we've shown that $\mathbf{A}^\top \mathbf{A}$ has eigenvalues σ_i^2 with corresponding eigenvectors $\mathbf{v}_i$. We also can also show that $\mathbf{A}\mathbf{A}^\top$ has eigenvalues σ_i^2, but with corresponding eigenvectors $\mathbf{u}_i$.

$$\mathbf{A}\mathbf{A}^\top \mathbf{u}_i = \sigma_i \mathbf{A}\mathbf{v}_i = \sigma_i^2 \mathbf{u}_i$$

Proposition 3.6. *Let $\mathbf{A}$ be any matrix of rank r. Then the non-zero eigenvalues of both $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}^\top\mathbf{A}$ are $\sigma_1^2, \ldots, \sigma_r^2$. The corresponding unit eigenvectors of $\mathbf{A}\mathbf{A}^\top$ are given by the columns of $\mathbf{U}$, and the corresponding unit eigenvectors of $\mathbf{A}^\top\mathbf{A}$ are given by the columns of $\mathbf{V}$.*

Notes:

1. The SVD expresses a matrix as a sum of rank-1 matrices

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top.$$

We can think of these as a list of the building blocks of $\mathbf{A}$ ordered by their importance ($\sigma_1 \geq \sigma_2 \geq \ldots$).

2. The singular value decomposition theorem shows that every matrix is diagonal, provided one uses the proper bases for the domain and range spaces. We can **diagonalize A** by

$$\mathbf{U}^\top \mathbf{A}\mathbf{V} = \mathbf{\Sigma}.$$

3. The SVD reveals a great deal about a matrix. Firstly, the rank of $\mathbf{A}$ is the number of non-zero singular values. The left singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are an orthonormal basis for the columns space of $\mathbf{A}$, $\mathcal{C}(\mathbf{A})$, and the right singular vectors $\mathbf{v}_1, \ldots, \mathbf{v}_r$ are an orthonormal basis for $\mathcal{C}(\mathbf{A}^\top)$, the row space of $\mathbf{A}$. The vectors $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_p$ from the non-compact SVD are a basis for the kernel of $\mathbf{A}$ (sometimes called the null space $\mathcal{N}(\mathbf{A})$), and $\mathbf{u}_{r+1}, \ldots, \mathbf{u}_n$ are a basis for $\mathcal{N}(\mathbf{A}^\top)$.

4. The SVD has many uses in mathematics. One is as a generalized inverse of a matrix. If $\mathbf{A}$ is $n \times p$ with $n \neq p$, or if it is square but not of full rank, then $\mathbf{A}$ cannot have an inverse. However, we say $\mathbf{A}^+$ is a generalized inverse if $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$. One such generalized inverse can be obtained from the SVD by $\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$ - this is known as the Moore-Penrose pseudo-inverse.

3.3.1 Examples

In practice, we don't compute SVDs of a matrix by hand: in R you can use the command `SVD(A)` to compute the SVD of matrix `A`. However, it is informative to do the calculation yourself a few times to help fix the ideas.

Example 3.2. Consider the matrix $\mathbf{A} = \mathbf{x}\mathbf{y}^\top$. We can see this is a rank-1 matrix, so it only has one non-zero singular value which is $\sigma_1 = ||\mathbf{x}||.||\mathbf{y}||$. Its SVD is given by

$$\mathbf{U} = \frac{1}{||\mathbf{x}||}\mathbf{x}, \quad \mathbf{V} = \frac{1}{||\mathbf{y}||}\mathbf{y}, \quad \text{and } \Sigma = ||\mathbf{x}||.||\mathbf{y}||.$$

Example 3.3. Let

$$\mathbf{A} = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}.$$

Let's try to find the SVD of $\mathbf{A}$.

We know the singular values are the square roots of the eigenvalues of $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}^\top\mathbf{A}$. We'll work with the former as it is only 2×2.

$$\mathbf{A}\mathbf{A}^\top = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix} \quad \text{and so } \det(\mathbf{A}\mathbf{A}^\top - \lambda\mathbf{I}) = (17-\lambda)^2 - 64$$

Solving $\det(\mathbf{A}\mathbf{A}^\top - \lambda\mathbf{I}) = 0$ gives the eigenvalues to be $\lambda = 25$ or 9. Thus the singular values of $\mathbf{A}$ are $\sigma_1 = \sqrt{25} = 5$ and $\sigma_2 = \sqrt{9} = 3$, and

$$\mathbf{\Sigma} = \begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix}.$$

The columns of $\mathbf{U}$ are the *unit* eigenvectors of $\mathbf{A}\mathbf{A}^\top$ which we can find by solving

$$(\mathbf{A}\mathbf{A}^\top - 25\mathbf{I}_2)\mathbf{u} = \begin{pmatrix} -8 & 8 \\ 8 & -8 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{and}$$
$$(\mathbf{A}\mathbf{A}^\top - 9\mathbf{I}_2)\mathbf{u} = \begin{pmatrix} 8 & 8 \\ 8 & 8 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

And so, remembering that the eigenvectors used to form $\mathbf{U}$ need to be *unit* vectors, we can see that

$$\mathbf{U} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Finally, to compute $\mathbf{V}$ recall that $\sigma_i\mathbf{v}_i = \mathbf{A}^\top\mathbf{u}_i$ and so

$$\mathbf{V} = \mathbf{A}^\top\mathbf{U}\mathbf{\Sigma}^{-1} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & \frac{1}{3} \\ 1 & \frac{-1}{3} \\ 0 & \frac{4}{3} \end{pmatrix}.$$

This completes the calculation, and we can see that we can express $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} 5 & 0 \\ 0 & 3 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{3\sqrt{2}} & \frac{-1}{3\sqrt{2}} & \frac{4}{3\sqrt{2}} \end{pmatrix}$$

or as the sum of rank-1 matrices:

$$\mathbf{A} = 5\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} \end{pmatrix} + 3\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \frac{1}{3\sqrt{2}} & \frac{-1}{3\sqrt{2}} & \frac{4}{3\sqrt{2}} \end{pmatrix}$$

This is the compact form of the SVD. To find the non-compact form we need $\mathbf{V}$ to be a 3×3 matrix, which requires us to find a 3rd column that is orthogonal to the first two columns (thus completing an orthonormal basis for $\mathbb{R}^3$). We can do that with the vector $\mathbf{v}_3 = \frac{1}{\sqrt{17}}(2 \ -2 \ -1)$ giving the non-compact SVD for $\mathbf{A}$.

$$\mathbf{A} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} & \frac{2}{\sqrt{17}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{3\sqrt{2}} & \frac{-2}{\sqrt{17}} \\ 0 & \frac{4}{3\sqrt{2}} & \frac{-1}{\sqrt{17}} \end{pmatrix}^\top$$

Let's check our answer in R.

```
A<- matrix(c(3,2,2,2,3,-2), nr=2, byrow=T)
svd(A)
```

```
## $d
## [1] 5 3
##
## $u
```

```
##             [,1]       [,2]
## [1,] -0.7071068 -0.7071068
## [2,] -0.7071068  0.7071068
##
## $v
##                [,1]       [,2]
## [1,] -7.071068e-01 -0.2357023
## [2,] -7.071068e-01  0.2357023
## [3,] -5.551115e-17 -0.9428090
```

The eigenvectors are only defined upto multiplication by -1 and so we can multiply any pair of left and right singular vectors by -1 and it is still a valid SVD.

Note: In practice this is a terrible way to compute the SVD as it is prone to numerical error. In practice an efficient iterative method is used in most software implementations (including R).

3.4 SVD optimization results

Why are eigenvalues and singular values useful in statistics? It is because they appear as the result of some important optimization problems. We'll see more about this in later chapters, but we'll prove a few preliminary results here.

For example, suppose $\mathbf{x} \in \mathbb{R}^n$ is a random variable with $\mathbb{C}\,\mathrm{ov}(\mathbf{x}) = \mathbf{\Sigma}$ (an $n \times n$ matrix), then can we find a projection of $\mathbf{x}$ that has either maximum or minimum variance? I.e., can we find $\mathbf{a}$ such that

$$\mathbb{V}\,\mathrm{ar}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}$$

is maximized or minimized? To make the question interesting we need to constrain the length of $\mathbf{a}$ so lets assume that $||\mathbf{a}||_2 = \sqrt{\mathbf{a}^\top \mathbf{a}} = 1$, otherwise we could just take $\mathbf{a} = \mathbf{0}$ to obtain a projection with variance zero. So we want to solve the optimization problems involving the quadratic form $\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}$:

$$\max_{\mathbf{a}:\, \mathbf{a}^\top \mathbf{a} = 1} \mathbf{A}^\top \mathbf{\Sigma} \mathbf{A}, \quad \text{and} \quad \min_{\mathbf{a}:\, \mathbf{a}^\top \mathbf{a} = 1} \mathbf{A}^\top \mathbf{\Sigma} \mathbf{A}. \tag{3.2}$$

Given that $\mathbf{\Sigma}$ is symmetric, we can write it as

$$\mathbf{\Sigma} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues of $\mathbf{\Sigma}$, and $\mathbf{V}$ is an orthogonal matrix of eigenvectors. If we let $\mathbf{b} = \mathbf{V}^\top \mathbf{a}$ then

$$\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a} = \mathbf{b}^\top \mathbf{\Lambda} \mathbf{b} = \sum_{i=1}^{n} \lambda_i b_i^2$$

and given that the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq \ldots$ and that

$$\sum_{i=1}^{n} b_i^2 = \mathbf{b}^\top \mathbf{b} = \mathbf{a}^\top \mathbf{V}\mathbf{V}^\top \mathbf{a} = \mathbf{a}^\top \mathbf{a} = 1,$$

we can see that the maximumn is λ_1 obtained by setting $\mathbf{b} = (1\ 0\ 0 \ldots)^\top$. Then

$$\begin{aligned} \mathbf{V}^\top \mathbf{a} &= \mathbf{b} \\ \mathbf{V}\mathbf{V}^\top \mathbf{a} &= \mathbf{V}\mathbf{b} \\ \mathbf{a} &= \mathbf{v}_1 \end{aligned}$$

so we can see that the maximum is obtained when $\mathbf{a} = \mathbf{v}_1$, the eigenvector of $\mathbf{\Sigma}$ corresponding to the largest eigenvalue λ_1.

Similarly, the minimum is λ_n, which obtained by setting $\mathbf{b} = (\ 0\ 0 \ldots\ 0\ 1)^\top$ which corresponds to $\mathbf{a} = \mathbf{v}_n$.

Proposition 3.7. *For any symmetric $n \times n$ matrix $\mathbf{\Sigma}$,*

$$\max_{\mathbf{a}:\mathbf{a}^\top \mathbf{a}=1} \mathbf{a}^\top \mathbf{\Sigma} \mathbf{a} = \lambda_1,$$

where the maximum occurs at $\mathbf{a} = \pm \mathbf{v}_1$, and

$$\min_{\mathbf{a}:\mathbf{a}^\top \mathbf{a}=1} \mathbf{a}^\top \mathbf{\Sigma} \mathbf{a} = \lambda_n$$

where the minimum occurs at $\mathbf{a} = \pm \mathbf{v}_n$, where $\lambda_i, \mathbf{v}_i$ are the ordered eigenpairs of $\mathbf{\Sigma}$.

Note that

$$\frac{\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}}{\mathbf{a}^\top \mathbf{a}} = \frac{\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}}{||\mathbf{a}||^2} = (\frac{\mathbf{a}}{||\mathbf{a}||})^\top \mathbf{\Sigma} (\frac{\mathbf{a}}{||\mathbf{a}||})$$

and so another way to write the maximization problems (3.2) is as unconstrained optimization problems:

$$\max_{\mathbf{a}} \frac{\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}}{\mathbf{a}^\top \mathbf{a}} \quad \text{and} \quad \min_{\mathbf{a}} \frac{\mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}}{\mathbf{a}^\top \mathbf{a}}.$$

We obtain a similar result for non-square matrices using the singular value decomposition.

Proposition 3.8. *For any matrix $\mathbf{A}$*

$$\max_{\mathbf{x}:||\mathbf{x}||_2=1} ||\mathbf{A}\mathbf{x}||_2 = \max_{\mathbf{x}} \frac{||\mathbf{A}\mathbf{x}||_2}{||\mathbf{x}||_2} = \sigma_1$$

the first singular value of $\mathbf{A}$, with the maximum achieved at $\mathbf{x} = \mathbf{v}_1$ (the first right singular vector).

Proof. This is follows from 3.7 as

$$||\mathbf{Ax}||_2^2 = \mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax}.$$

□

Here, $||\mathbf{x}|_2$ denotes the Euclidean norm for vectors:

$$||\mathbf{x}||_2 = (\mathbf{x}^\top \mathbf{x})^{\frac{1}{2}}.$$

Finally, we will need the following result when we study canonical correlation analysis:

Proposition 3.9. *For any matrix* $\mathbf{A}$*, we have*

$$\max_{\mathbf{a},\mathbf{b}:\, ||\mathbf{a}||=||\mathbf{b}||=1} \mathbf{a}^\top \mathbf{Ab} = \sigma_1.$$

with the maximum obtained at $\mathbf{a} = \mathbf{u}_1$ *and* $\mathbf{b} = \mathbf{v}_1$*, the first left and right singular vectors of* $\mathbf{A}$*.*

Proof. See Section 5.1 □

We'll see much more of this kind of thing in Chapters 4 and 5.

3.5 Low-rank approximation

One of the reasons the SVD is so widely used is that it can be used to find the best low rank approximation to a matrix. Before we discuss this, we need to define what it means for some matrix $\mathbf{B}$ to be a good approximation to $\mathbf{A}$. To do that, we need the concept of a matrix norm.

3.5.1 Matrix norms

In Section 2.3.1 we described norms on vectors. Here will extend this idea to include norms on matrices, so that we can discuss the size of a matrix $||\mathbf{A}||$, and the distance between two matrices $||\mathbf{A} - \mathbf{B}||$. There are two particular norms we will focus on. The first is called the Frobenious norm (or sometimes the Hilbert-Schmidt norm).

Definition 3.1. Let $\mathbf{A} \in \mathbb{R}^{n \times p}$. The **Frobenius norm** of $\mathbf{A}$ is

$$||\mathbf{A}||_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{p} |a_{ij}|^2 \right)^{\frac{1}{2}} = (\mathrm{tr}\, \mathbf{A}^\top \mathbf{A})^{\frac{1}{2}}$$

where a_{ij} are the individual entries of $\mathbf{A}$.

Note that the Frobenius norm is invariant to rotation by an orthogonal matrix $\mathbf{U}$:

$$\begin{aligned} ||\mathbf{AU}||_F^2 &= \text{tr}(\mathbf{U}^\top \mathbf{A}^\top \mathbf{AU}) \\ &= \text{tr}(\mathbf{UU}^\top \mathbf{A}^\top \mathbf{A}) \\ &= \text{tr}(\mathbf{A}^\top \mathbf{A}) \\ &= ||\mathbf{A}||_F^2. \end{aligned}$$

Proposition 3.10.

$$||\mathbf{A}||_F = \left(\sum_{i=1}^{r} \sigma_i^2 \right)^{\frac{1}{2}}$$

where σ_i are the singular values of $\mathbf{A}$, and $r = \text{rank}(\mathbf{A})$.

Proof. Using the (non-compact) SVD $\mathbf{A} = \mathbf{U\Sigma V}^\top$ we have

$$||\mathbf{A}||_F = ||\mathbf{U}^\top \mathbf{A}||_F = ||\mathbf{U}^\top \mathbf{AV}||_F = ||\mathbf{\Sigma}||_F = \text{tr}(\mathbf{\Sigma}^\top \mathbf{\Sigma})^{\frac{1}{2}} = \left(\sum \sigma_i^2 \right)^{\frac{1}{2}}.$$

□

We previously defined the p-norms for vectors in $\mathbb{R}^p$ to be

$$||\mathbf{x}||_p = \left(\sum |x_i|^p \right)^{\frac{1}{p}}.$$

These vector norms *induce* matrix norms, sometimes also called operator norms:

Definition 3.2. The p-norms for matrices are defined by

$$||\mathbf{A}||_p = \sup_{\mathbf{x} \neq 0} \frac{||\mathbf{Ax}||_p}{||\mathbf{x}||_p} = \sup_{\mathbf{x}: ||\mathbf{x}||_p = 1} ||\mathbf{Ax}||_p$$

Proposition 3.11.

$$||\mathbf{A}||_2 = \sigma_1$$

where σ_1 is the first singular value of $\mathbf{A}$.

Proof. By Proposition 3.8. □

3.5.2 Eckart-Young-Mirsky Theorem

Now that we have defined a norm (i.e., a distance) on matrices, we can think about approximating a matrix $\mathbf{A}$ by a matrix that is easier to work with. We have shown that any matrix can be split into the sum of rank-1 component matrices

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

We'll now consider a family of approximations of the form

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \tag{3.3}$$

where $k <= r = \text{rank}(\mathbf{A})$. This is a rank-k matrix, and as we'll now show, it is the best possible rank-k approximation to $\mathbf{A}$.

Theorem 3.1. ***(Eckart-Young-Mirsky)*** *For either the 2-norm $||\cdot||_2$ or the Frobenious norm $||\cdot||_F$*

$$||\mathbf{A} - \mathbf{A}_k|| \leq ||\mathbf{A} - \mathbf{B}|| \text{ for all rank-k matrices } \mathbf{B}.$$

Moreover,

$$||\mathbf{A} - \mathbf{A}_k|| = \begin{cases} \sigma_{k+1} & \text{for the } ||\cdot||_2 \text{ norm} \\ \left(\sum_{i=k+1}^{r} \sigma_i^2\right)^{\frac{1}{2}} & \text{for the } ||\cdot||_F \text{ norm.} \end{cases}$$

Proof. The last part follows from Propositions 3.11 and 3.10.

Non-examinable: this is quite a tricky proof, but I've included it as its interesting to see. We'll just prove it for the 2-norm. Let $\mathbf{B}$ be an $n \times p$ matrix of rank k. The null space $\mathcal{N}(\mathbf{B}) \subset \mathbb{R}^p$ must be of dimension $p - k$ by the rank nullity theorem.

Consider the $p \times (k+1)$ matrix $\mathbf{V}_{k+1} = [\mathbf{v}_1 \ \ldots \ \mathbf{v}_{k+1}]$. This has rank $k+1$, and has column space $\mathcal{C}(\mathbf{V}_{k+1}) \subset \mathbb{R}^p$. Because

$$\dim \mathcal{N}(\mathbf{B}) + \dim \mathcal{C}(\mathbf{V}_{k+1}) = p - k + k + 1 = p + 1$$

we can see that $\mathcal{N}(B)$ and $\mathcal{C}(V_{k+1})$ cannot be disjoint spaces (as they are both subsets of the p-dimensional space $\mathbb{R}^p$). Thus we can find $\mathbf{w} \in \mathcal{N}(B) \cap \mathcal{C}(V_{k+1})$, and moreover we can choose $\mathbf{w}$ so that $||\mathbf{w}||_2 = 1$.

Because $\mathbf{w} \in \mathcal{C}(\mathbf{V}_{k+1})$ we can write $\mathbf{w} = \sum_{i=1}^{k+1} w_i \mathbf{v}_i$ with $\sum_{i=1}^{k+1} w_i^2 = 1$.

Then

$$
\begin{aligned}
||\mathbf{A}-\mathbf{B}||_2^2 &\geq ||(\mathbf{A}-\mathbf{B})\mathbf{w}||_2^2 \quad \text{by definition of the matrix 2-norm} \\
&= ||\mathbf{A}\mathbf{w}||_2^2 \quad \text{as } \mathbf{w} \in \mathcal{N}(\mathbf{B}) \\
&= \mathbf{w}^\top \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\mathbf{w} \quad \text{using the SVD} \mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \\
&= \sum_{i=1}^{k+1} \sigma_i^2 w_i^2 \quad \text{by substituting } \mathbf{w} = \sum_{i=1}^{k+1} w_i \mathbf{v}_i \\
&\geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} w_i^2 \quad \text{as } \sigma_1 \geq \sigma_2 \geq \ldots \\
&= \sigma_{k+1}^2 \quad \text{as } \sum_{i=1}^{k+1} w_i^2 = 1 \\
&= ||\mathbf{A}-\mathbf{A}_k||_2^2
\end{aligned}
$$

as required □

This best-approximation property is what makes the SVD so useful in applications.

3.5.3 Example: image compression

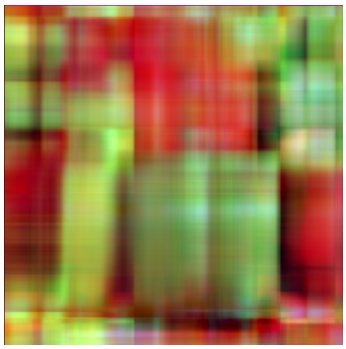
As an example, lets consider the image of some peppers from the USC-SIPI image database.

```
library(tiff)
library(rasterImage)
peppers<-readTIFF("figs/Peppers.tiff")
plot(as.raster(peppers))
```

This is a 512×512 colour image, meaning that there are three matrices $\mathbf{R}, \mathbf{B}, \mathbf{G}$ of dimension 512×512) giving the intensity of red, green, and blue for each pixel. Naively storing this matrix requires 5.7Mb.

We can compute the SVD of the three colour intensity matrices, and the view the image that results from using reduced rank versions $\mathbf{B}_k, \mathbf{G}_k, \mathbf{R}_k$ instead (as in Equation (3.3)). The image below is formed using $k = 5, 30, 100$, and 300 basis vectors.

```
svd_image <- function(im,k){
  s <- svd(im)
  Sigma_k <- diag(s$d[1:k])
  U_k <- s$u[,1:k]
  V_k <- s$v[,1:k]
  im_k <- U_k %*% Sigma_k %*% t(V_k)
   ## the reduced rank SVD produces some intensities <0 and >1.
  # Let's truncate these
  im_k[im_k>1]=1
  im_k[im_k<0]=0
  return(im_k)
}

par(mfrow=c(2,2), mar=c(1,1,1,1))

pepprssvd<- peppers
for(k in c(4,30,100,300)){
  svds<-list()
  for(ii in 1:3) {
    pepprssvd[,,ii]<-svd_image(peppers[,,ii],k)
  }
  plot(as.raster(pepprssvd))
}
```




You can see that for $k = 30$ we have a reasonable approximation, but with some errors. With $k = 100$ it is hard to spot the difference with the original. The size of the four compressed images is 45Kb, 345Kb, 1.1Mb and 3.4Mb.

You can see further demonstrations of image compression with the SVD here.

We will see much more of the SVD in later chapters.

3.6 Computer tasks

1. Finding the eigenvalues and eigenvectors of a matrix is easy in R.

```
A=matrix(c(3,1,1,6),nrow=2,byrow=TRUE)    # use a to define a matrix A
Eig=eigen(A)                             # the eigenvalues and eigenvectors of A
                                           # are stored in the list Eig
lambda=Eig$values                        # extract the eigenvalues from Eig and
                                           # store in the vector e
lambda                                   # you should see the eigenvalues in
```

```
## [1] 6.302776 2.697224
```

```
                                           # descending order
Q=Eig$vectors                            # extract the eigenvectors from Eig and
```

```
                                    # store then in the columns of Q
```

The spectral decomposition of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$$

Let's check this in R (noting as always that there may be some numerical errors)

```
Q%*%diag(lambda)%*%t(Q)          # reconstruct A,
```

```
##      [,1] [,2]
## [1,]    3    1
## [2,]    1    6
```

```
                                    # where t(Q) gives the transpose of Q
```

Since A is positive definite, we can calculate the symmetric, positive definite square root of A.

```
Asqrt=Q%*%diag(lambda**0.5)%*%t(Q) # lambda**0.5 contains the square roots
Asqrt%*%Asqrt                      # it is seen that A is recovered
```

```
##      [,1] [,2]
## [1,]    3    1
## [2,]    1    6
```

- Instead of using the full eigendecomposition for $\mathbf{A}$, try truncating it and using just a single eigenvalue and eigenvector, i.e., compute

$$\mathbf{A}' = \lambda_1 \mathbf{q}_1 \mathbf{q}_1^\top$$

- Compute the difference between $\mathbf{A}$ and $\mathbf{A}'$ using the 2-norm and the Frobenius norm.

2. The singular value decomposition can be computed in R using the command `svd`. Let $\mathbf{X}$ be the four numerical variables in the `iris` dataset with the column mean removed

```
n=150
H=diag(rep(1,n))-rep(1,n)%*%t(rep(1,n))/n   # calculate the centering matrix H
X=H%*% as.matrix(iris[,1:4])
# This can also be done using the command
# sweep(iris[,1:4], 2, colMeans(iris[,1:4]))  # do you understand why?
```

- Compute the SVD of $\mathbf{X}$ in R and report its singular values.
- Does R report the full or compact SVD?
- Check that $\mathbf{X}\mathbf{v} = \sigma\mathbf{u}$.
- Compute the best rank-1, rank-2, and rank-3 approximations to $\mathbf{X}$, and report the 2-norm and Frobenious norm for these approximations

- Compute the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. How do these relate to the singular values? How does $\mathbf{X}^\top\mathbf{X}$ relate to the sample covariance matrix of the iris data? How do the singular values relate to the eigenvalues of the covariance matrix?

- Let $\mathbf{S}$ be the sample covariance matrix of the iris dataset. What vector $\mathbf{x}$ with $||\mathbf{x}|| = 1$ maximizes $\mathbf{x}^\top\mathbf{S}\mathbf{x}$?

3. Choose an few images from the USC-SIPI Image Database and repeat the image compression example from the notes. Which type of images compress well do you think?

4. We won't discuss how the SVD is computed in practice, but there are a variety of approaches that can be used. Try the following iterative approach for computing the first singular vectors:

```
X <- as.matrix(iris[,1:4])
v <- rnorm(dim(X)[2])

for (iter in 1:500){
  u <- X %*%v
  u <- u/sqrt(sum(u^2))
  v <- t(X) %*%u
  v <- v/sqrt(sum(v^2))
}
X.svd <- svd(X)
X.svd$v[,1]/v
X.svd$u[,1]/u
```

3.7 Exercises

0. There is a great Twitter thread on the SVD by Daniella Witten. Read it here.

1. Let $\mathbf{\Sigma}$ be an arbitrary covariance matrix.

   - Show $\mathbf{\Sigma}$ is symmetric and positive semi-definite.
   - Give examples of both singular and non-singular covariance matrices.

   - What condition must the eigenvalues of a non-singular covariance matrix satisfy?

2. Compute, by hand (but check your answer in R), the singular value decomposition (full and compact) of the following matrices.
   - $\begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix}$

- $\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$

3. Let

$$\mathbf{X} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The eigen-decomposition of $\mathbf{X}^\top\mathbf{X}$ is

$$\mathbf{X}^\top\mathbf{X} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}^\top$$

Use this fact to compute answer the following questions:

- What are the singular values of $\mathbf{X}$?
- What are the right singular vectors of $\mathbf{X}$?
- What are the left singular vectors of $\mathbf{X}$?
- Give the compact SVD of $\mathbf{X}$. Check your answer, noting that the singular vectors are only specified up to multiplication by -1
- Can you compute the full SVD of $\mathbf{X}$?
- What is the eigen-decomposition of $\mathbf{X}\mathbf{X}^\top$?
- Find a generalised inverse of matrix $\mathbf{X}$.

4. The SVD can be used to solve linear systems of the form

$$\mathbf{A}\mathbf{x} = \mathbf{y}$$

where $\mathbf{A}$ is a $n \times p$ matrix, with compact SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$.

- If $\mathbf{A}$ is a square invertible matrix, show that

$$\tilde{\mathbf{x}} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{y}$$

is the unique solution to $\mathbf{A}\mathbf{x} = \mathbf{y}$, i.e., show that $\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$.

- If $\mathbf{A}$ is not a square matrix, then $\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$ is a generalized inverse (not a true inverse) matrix, and $\tilde{\mathbf{x}} = \mathbf{A}^+\mathbf{y}$ is still a useful quantity to consider as we shall now see. Let $\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\mathbf{y} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$. Then $\mathbf{A}\mathbf{x} = \mathbf{y}$ is an over-determined system in that there are 3 equations in 2 unknowns. Compute $\tilde{\mathbf{x}} = \mathbf{A}^+\mathbf{y}$. Is this a solution to the equation?

Note that you computed the svd for $\mathbf{A}$ in Q2.

- Now suppose $\mathbf{y} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$. There is no solution to $\mathbf{Ax} = \mathbf{y}$ in this case as $\mathbf{y}$ is not in the column space of $\mathbf{A}$. Prove that $\tilde{\mathbf{x}} = \mathbf{A}^{+}\mathbf{y}$ solves the least squares problem

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} ||\mathbf{y} - \mathbf{Ax}||_2.$$

Hint: You can either do this directly for this problem, or you can show that the least squares solution $(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{y} = \tilde{\mathbf{x}}$.

5. Consider the system

$$\mathbf{Bx} = \mathbf{y} \text{ with } \mathbf{B} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

This is an underdetermined system, as there are 2 equations in 3 unknowns, and so there are an infinite number of solutions for $\mathbf{x}$ in this case.

- Find the full SVD for $\mathbf{B} = \mathbf{U\Sigma V}^\top$ (noting that $\mathbf{B} = \mathbf{A}^\top$ for $\mathbf{A}$ from the previous question).
- Compute $\tilde{\mathbf{x}} = \mathbf{B}^{+}\mathbf{y}$, check it is a solution to the equation, and explain why

$$\tilde{\mathbf{x}} = \sum_{i=1}^{r} \mathbf{v}_i \frac{\mathbf{u}_i^\top \mathbf{y}}{\sigma_i}$$

in general, where $r \leq max(n, p)$ is the rank of $\mathbf{B}$, and write out $\tilde{\mathbf{x}}$ explicitly in this form for the given $\mathbf{B}$.
- Consider $\mathbf{x}$ of the form

$$\mathbf{x} = \tilde{\mathbf{x}} + \sum_{i=r+1}^{n} \alpha_i \mathbf{v}_i$$

and explain why any $\mathbf{x}$ of this form is also a solution to $\mathbf{Bx} = \mathbf{y}$. Thus write out all possible solutions of the equation.
- Prove that $\tilde{\mathbf{x}}$ is the solution with minimum norm, i.e., $||\tilde{\mathbf{x}}||_2 \leq ||\mathbf{x}||_2$. **Hint** $\mathbf{v}_1, \ldots, \mathbf{v}_p$ form a complete orthonormal basis for $\mathbb{R}^p$.

6. Prove proposition 3.4, namely that the eigenvalues of projection matrices are either 0 or 1. Show that $\mathbf{1}_n$ is an eigenvector of $\mathbf{H}$. What is the corresponding eigenvalue? What are the remaining eigenvalues equal to?

PART II: Dimension reduction methods

Introductory Video

In many applications, a large number of variables are recorded for each experimental unit under study. For example, if we think of individual people as the *experimental units*, then in a health check-up we might collect data on age, blood pressure, cholesterol level, blood test results, lung function, weight, height, BMI, etc. If you use websites such as Amazon, Facebook, and Google, they store thousands (possibly millions) of pieces of information about you (this article shows you how to download the information Google stores about you, including all the locations you've visited, every search, youtube video, or app you've used and more). They process this data to create an individual profile for each user, which they can then use to create targetted adverts.

When analysing data of moderate or high dimension, it is often desirable to seeks ways to restructure the data and reduce its dimension whilst **retaining the most important information** within the data or **preserving some feature of interest** in the data. There a variety of reasons we might want to do this.

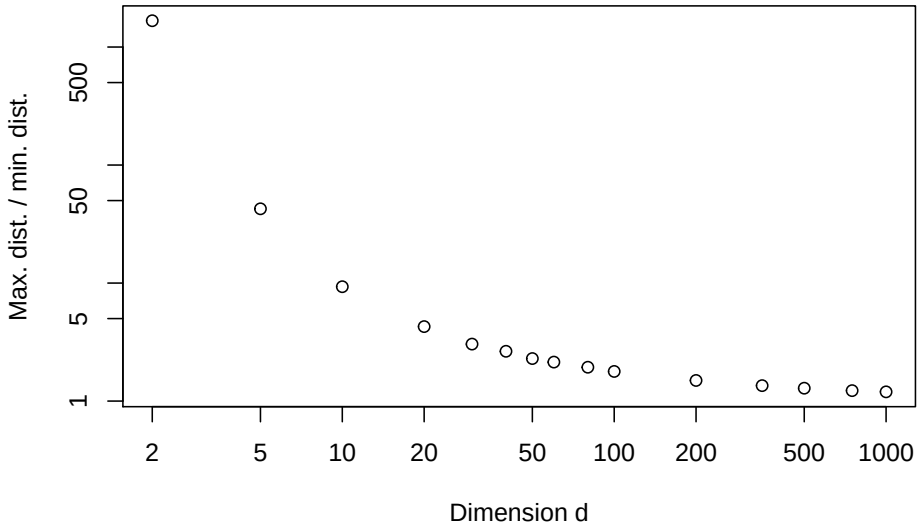
- In reduced dimensions, it is often much easier to understand and appreciate the most important features of a dataset.
- If there is a lot of reduncancy in the data, we might want to reduce the dimension to lower the memory requirements in storing it (e.g. with sound and image compression).
- In high dimensions, it can be difficult to analyse data (e.g. with statistical methods), and so reducing the dimension can be a way to make a dataset amenable to analysis.

In this part of the module we investigate three different methods for dimension reduction: Principal Component Analysis (PCA) in Chapter 4; Canonical Correlation Analysis (CCA) in Chapter 5; and Multidimensional Scaling (MDS) in Chapter 6. Matrix algebra (Chapters 2 and 3) plays a key role in all three of these techniques.

A warning

Beware that high-dimensional data can behave qualitatively differently to low-dimensional data. As an example, lets consider 1000 points uniformly distributed in $[0,1]^d$, and think about how close together or spread out the points are. A simple way to do this is to consider the ratio of the maximum and minimum distance between any two points in our sample.

```
N<-1000
averatio <-c()
ii<-1
for(d in c(2,5,10,20,30,40,50,60,80,100, 200, 350, 500, 750, 1000)){
  averatio[ii] <- mean(replicate(10, {
  X<-matrix(runif(N*d), nc=d)
  d <- as.matrix(dist(X))
  # this gives a N x N matrix of the Euclidean distances between the data points.
  maxdist <- max(d)
  mindist <- min(d+diag(10^5, nrow=N))
  # The diagonal elements of the distance matrix are zero,
  # so I've added a big number to the diagonal
  # so that we get the minimum distance between different points
  maxdist/mindist}))
  ii <- ii+1
}
plot(c(2,5,10,20,30,40,50,60,80,100, 200, 350, 500, 750, 1000),
     averatio, ylab='Max. dist. / min. dist.', xlab='Dimension d', log='xy')
```



So we can see that as the dimension increases, the ratio of the maximum and minimum distance between any two random points in our sample tends to 1. In other words, all points are the same distance apart!

Chapter 4

Principal Component Analysis (PCA)

The videos for this chapter are available at the following links:

- 4.1 An informal introduction to PCA
- 4.1.5 Informal Examples
- 4.2 A more formal description of PCA
- 4.2.1 Properties of PC scores
- 4.2.2 Example: PCA of the football data
- 4.2.4 Population PCA and transformations
- 4.3 An alternative view of PCA: minimizing reconstruction error‘
- 4.3.1 Example: PCA of the MNIST data

With multivariate data, it is common to want to reduce the dimension of the data *in a sensible way*. For example

- exam marks across different modules are averaged to produce a single overall mark for each student
- a football league table converts the numbers of wins, draws and losses to a single measure of points.

Mathematically, these summaries are both linear combinations of the original variables of the form

$$y = \mathbf{u}^\top \mathbf{x}.$$

for some choice of $\mathbf{u}$.

For the exam marks example, suppose each student sits $p = 4$ modules with marks, x_1, x_2, x_3, x_4. Then, writing $\mathbf{x} = (x_1, x_2, x_3, x_4)^\top$ and choosing $\mathbf{u} = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)^\top$

gives an overall average,

$$y = \mathbf{u}^\top \mathbf{x} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \frac{x_1}{4} + \frac{x_2}{4} + \frac{x_3}{4} + \frac{x_4}{4}.$$

For the football league table, if w is the number of wins, d is the number of draws and l is the number of losses then, writing $\mathbf{r} = (w, d, l)^\top$, we choose $\mathbf{u} = (3, 1, 0)^\top$ to get the points score

$$y = \mathbf{u}^\top \mathbf{r} = \begin{pmatrix} 3 & 1 & 0 \end{pmatrix} \begin{pmatrix} w \\ d \\ l \end{pmatrix} = 3w + 1d + 0l = 3w + d.$$

Geometric interpretation

In the two examples above, we used the vector $\mathbf{u}$ to convert our original variables, $\mathbf{x}$, to a new variable, y, by projecting $\mathbf{x}$ onto $\mathbf{u}$. We can think of this as a projection onto the subspace defined by $\mathbf{u}$

$$U = \text{span}\{\mathbf{u}\} = \{\lambda \mathbf{u} : \lambda \in \mathbb{R}\} \subset \mathbb{R}^p,$$

For the exam data, each data point $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$ is a vector in $\mathbb{R}^4$, and we've expressed $\mathbf{x}$ in terms of its coordinates with respsect to the standard basis, $\mathbf{e}_1^\top = (1\ 0\ 0\ 0)$ etc:

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + x_3 \mathbf{e}_3 + x_4 \mathbf{e}_4.$$

The vector subspace U is a line in $\mathbb{R}^4$ along the direction $\mathbf{u} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}^\top$.

How do we project onto subspace U?

- If $||\mathbf{u}||_2 = 1$ then the orthogonal projection of $\mathbf{x}$ onto U is

  $$\mathbf{u}\mathbf{u}^\top \mathbf{x}.$$

  Or in other words, the projection of $\mathbf{x}$ onto subspace U has coordinate $\mathbf{u}^\top \mathbf{x}$ with respect to basis $\{\mathbf{u}\}$.

If you prefer to think in terms of projection matrices (see Chapter 2.3.3.1), then the matrix for projecting onto U is

$$\mathbf{P}_U = \mathbf{u}(\mathbf{u}^\top \mathbf{u})^{-1} \mathbf{u}^\top$$

which simplifies to

$$\mathbf{P}_U = \mathbf{u}\mathbf{u}^\top$$

when $||\mathbf{u}|| = \sqrt{\mathbf{u}^\top \mathbf{u}} = 1$ so that we again see the projection of $\mathbf{x}$ onto U is $y = \mathbf{P}_u \mathbf{x} = \mathbf{u}\mathbf{u}^\top \mathbf{x}$.

How should we choose u?

The answer to that question depends upon the goal of the analysis. For the exam and football league examples, the choice of $\mathbf{u}$ is an arbitrary decision taken in order to reduce a multidimensional dataset to a single variable (average mark, or points).

A single $\mathbf{u}$ gives a **snapshot** or summary of the data. If $\mathbf{u}$ is chosen well that snapshot may tell us much of what we want to know about the data, e.g.,

- Liverpool won the league,
- student X's exam performance was first class etc.

In many cases we will want to use multiple snapshots: instead of using a single $\mathbf{u}$, we will use a collection $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ and consider the derived variables

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^\top \mathbf{x} \\ \mathbf{u}_2^\top \mathbf{x} \\ \vdots \\ \mathbf{u}_r^\top \mathbf{x} \end{pmatrix}$$

In matrix notation, if we set

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \ldots & \mathbf{u}_r \\ | & & | \end{pmatrix}$$

then the new derived variable is

$$\mathbf{y} = \mathbf{U}^\top \mathbf{x}.$$

If $\dim(\mathbf{y}) = r < p = \dim(\mathbf{x})$ then we have reduced the dimension of the data. If $\mathbf{y}$ tells us all we need to know about the data, then we can work (plot, analyse, model) with $\mathbf{y}$ instead of $\mathbf{x}$. If $r \ll p$ this can make working with the data significantly easier, as we can more easily visulise and understand low dimensional problems.

We will study a variety of methods for choosing $\mathbf{U}$. The methods can all be expressed as constrained optimization problems:

$$\text{minimize} f_{\mathbf{X}}(\mathbf{U}) \tag{4.1}$$
$$\text{subject to } \mathbf{U} \in \mathcal{U} \tag{4.2}$$

The objective $f_{\mathbf{X}}(\mathbf{U})$ varies between methods: principal component analysis (PCA) maximizes variance or minimizes reconstruction error; canonical correlation analysis (CCA) maximizes correlation; multidimensional scaling (MDS) maximizes spread etc.

The contstraint on the search space $\mathcal{U}$, is usually that $\mathbf{U}$ must be (partially) orthogonal, but in other methods other constraints are used

4.1 PCA: an informal introduction

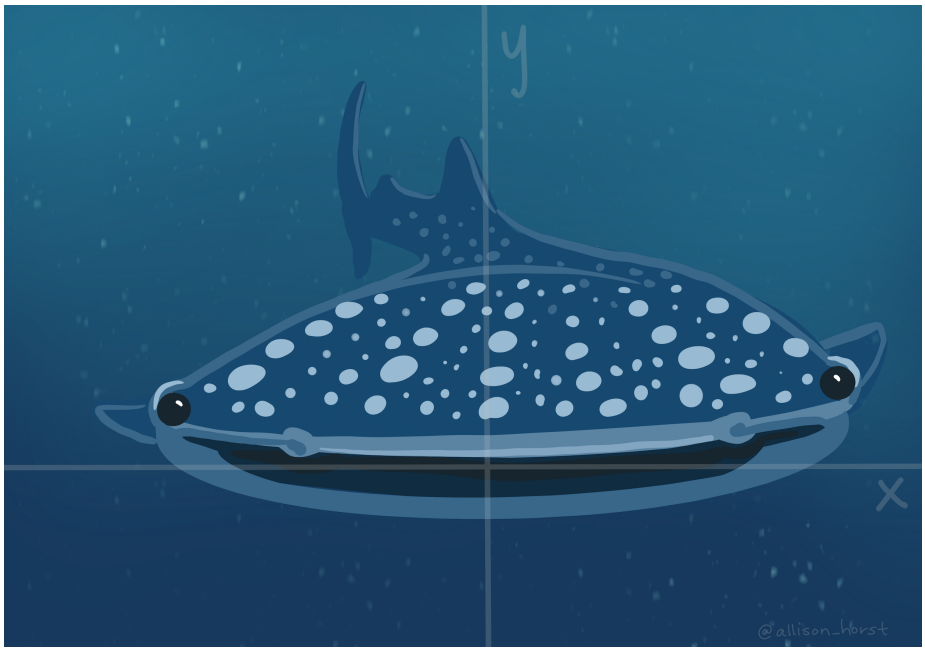
There are two different ways of motivating principal component analysis (PCA), which may in part explain why PCA is so widely used.

The first motivation, and the topic of this section, is to introduce PCA as method for maximizing the variance of the transformed variables $\mathbf{y}$. We start by choosing $\mathbf{u}_1$ so that $y_1 = \mathbf{u}_1^\top \mathbf{x}$ has maximum variance. We then choose $\mathbf{u}_2$ so that $y_2 = \mathbf{u}_2^\top \mathbf{x}$ has maximum variance subject to being uncorrelated with y_1, and so on.

The idea is to produce a set of variables $y_1, y_2, \ldots, y_r$ that are uncorrelated, but which are most informative about the data. The thinking is that if a variable has large variance it must be informative/important.

The name **principal component analysis** comes from thinking of this as splitting the data $\mathbf{X}$ into its most important parts. It therefore won't surprise you to find that this involves the matrix decompositions we studied in Chapter 3.

Allison Horst (@allison_horst) gave a great illustration of how to think about PCA on Twitter. Imagine you are a whale shark with a wide mouth



and that you're swimming towards a delicious swarm of krill.

What way should you tilt your shark head in order to eat as many krill as possible? The answer is given by the first principal component of the data!

4.1.1 Notation recap

As before, let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be $p \times 1$ vectors of measurements on n experimental units and write

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_1^\top & - \\ - & \mathbf{x}_2^\top & - \\ - & \vdots & - \\ - & \mathbf{x}_n^\top & - \end{pmatrix}$$

IMPORTANT NOTE: In this section we will assume that $\mathbf{X}$ has been column centered so that the mean of each column is 0 (i.e., the sample mean of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is the zero vector $\mathbf{0} \in \mathbb{R}^p$). If $\mathbf{X}$ has not been column centered, replace $\mathbf{X}$ by

$$\mathbf{HX}$$

where $\mathbf{H}$ is the centering matrix (see 2.4), or equivalently, replace $\mathbf{x}_i$ by $\mathbf{x}_i - \bar{\mathbf{x}}$. It is possible to write out the details of PCA replacing $\mathbf{X}$ by $\mathbf{HX}$ throughout, but this gets messy and obscures the important detail. Most software implementations (and in particular `prcomp` in R), automatically centre your data for you, and so in practice you don't need to worry about doing this when using a software package.

The sample covariance matrix for $\mathbf{X}$ (assuming it has been column centered) is

$$\mathbf{S} = \frac{1}{n}\mathbf{X}^\top\mathbf{X} = \frac{1}{n}\sum \mathbf{x}_i\mathbf{x}_i^\top$$

Given some vector $\mathbf{u}$, the transformed variables

$$y_i = \mathbf{u}^\top\mathbf{x}_i$$

have

- **mean** 0:
$$\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i = \frac{1}{n}\sum_{i=1}^n \mathbf{u}^\top\mathbf{x}_i = \frac{1}{n}\mathbf{u}^\top\sum_{i=1}^n \mathbf{x}_i = 0$$
as the mean of the $\mathbf{x}_i$ is $\mathbf{0}$.

- **sample covariance matrix**
$$\mathbf{u}^\top\mathbf{S}\mathbf{u}$$
as
$$\frac{1}{n}\sum_{i=1}^n y_i^2 = \frac{1}{n}\sum_{i=1}^n \mathbf{u}^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{u} = \frac{1}{n}\mathbf{u}^\top\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top\mathbf{u} = \mathbf{u}^\top\mathbf{S}\mathbf{u}$$

4.1.2 First principal component

We would like to find the $\mathbf{u}$ which maximises the sample variance, $\mathbf{u}^\top \mathbf{S}\mathbf{u}$ over unit vectors $\mathbf{u}$, i.e., vectors with $||\mathbf{u}|| = 1$. Why do we focus on unit vectors? If we don't, we could make the variance as large as we like, e.g., if we replace $\mathbf{u}$ by $10\mathbf{u}$ it would increase the variance by a factor of 100. Thus, we constrain the problem and only consider unit vectors for $\mathbf{u}$.

We know from Proposition 3.7 in Section 3.4 that $\mathbf{v}_1$, the first eigenvector of $\mathbf{S}$ (also the first right singular vector of $\mathbf{X}$), maximizes $\mathbf{u}^\top \mathbf{S}\mathbf{u}$ with

$$\max_{\mathbf{u}:||\mathbf{u}||=1} \mathbf{u}^\top \mathbf{S}\mathbf{u} = \mathbf{v}_1 \mathbf{S}\mathbf{v}_1 = \lambda_1$$

where λ_1 is the largest eigenvalue of $\mathbf{S}$.

So the first principal component of $\mathbf{X}$ is $\mathbf{v}_1$, and the first transformed variable (sometimes called a principal component score) is $y_1 = \mathbf{v}_1^\top \mathbf{x}$. Applying this to each data point we get n instances of this new variable

$$y_{i1} = \mathbf{v}_1^\top \mathbf{x}_i.$$

A note on singular values: We know $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$ and so the eigenvalues of $\mathbf{S}$ are the same as the squared singular values of $\frac{1}{\sqrt{n}}\mathbf{X}$:

$$\sqrt{\lambda_1} = \sigma_1\left(\frac{1}{\sqrt{n}}\mathbf{X}\right)$$

If we scale $\mathbf{X}$ by a factor c, then the singular values are scaled by the same amount, i.e.,

$$\sigma_i(c\mathbf{X}) = c\sigma_i(\mathbf{X})$$

and in particular

$$\sigma_i\left(\frac{1}{\sqrt{n}}\mathbf{X}\right) = \frac{1}{\sqrt{n}}\sigma_i(\mathbf{X})$$

We will need to remember this scaling if we use the SVD of $\mathbf{X}$ to do PCA. Note that scaling $\mathbf{X}$ does not change the singular vectors/principal components.

4.1.3 Second principal component

y_1 is the transformed variable that has maximum variance. What should we choose to be our next transformed variable, i.e., what $\mathbf{u}_2$ should we choose for $y_2 = \mathbf{u}_2^\top \mathbf{x}$? It makes sense to choose y_2 to be uncorrelated with y_1, as otherwise it contains some of the same information given by y_1. The sample covariance

between y_1 and $\mathbf{u}_2^\top \mathbf{x}$ is

$$\begin{aligned} s_{y_2 y_1} &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{u}_2^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{v}_1 \\ &= \mathbf{u}_2^\top \mathbf{S} \mathbf{v}_1 \\ &= \lambda_1 \mathbf{u}_2^\top \mathbf{v}_1 \text{ as } \mathbf{v}_1 \text{ is an eigenvector of } S \end{aligned}$$

So to make y_2 uncorrelated with y_1 we have to choose $\mathbf{u}_2$ to be orthogonal to $\mathbf{v}_1$, i.e., $\mathbf{u}_2^\top \mathbf{v}_1 = 0$. So we choose $\mathbf{u}_2$ to be the solution to the optimization problem

$$\max_{\mathbf{u}} \mathbf{u}^\top \mathbf{S} \mathbf{u} \text{ subject to } \mathbf{u}^\top \mathbf{v}_1 = 0.$$

The solution to this problem is to take $\mathbf{u}_2 = \mathbf{v}_2$, i.e., the second eigenvector of $\mathbf{S}$ (or second right singular vector of $\mathbf{X}$), and then

$$\mathbf{v}_2^\top \mathbf{S} \mathbf{v}_2 = \lambda_2.$$

We'll prove this result in the next section.

Later principal components

Our first transformed variable is

$$y_{i1} = \mathbf{v}_1^\top \mathbf{x}_i$$

and our second transformed variable is

$$y_{i2} = \mathbf{v}_2^\top \mathbf{x}_i.$$

At this point, you can probably guess that the j^{th} transformed variable is going to be

$$y_{ij} = \mathbf{v}_j^\top \mathbf{x}_i.$$

where $\mathbf{v}_j$ is the j^{th} eigenvector of $\mathbf{S}$.

- The transformed variables y_i are the **principal component scores**. y_1 is the first score etc.
- The eigenvectors/right singular vectors are sometimes refered to as the **loadings** or simply as the **principal components**.

4.1.4 Geometric interpretation

We think of PCA as projecting the data points $\mathbf{x}$ onto a subspace V. The basis vectors for this subspace are the eigenvectors of $\mathbf{S}$, which are the same as the right singular vectors of $\mathbf{X}$ (the loadings):

$$V = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_r\}.$$

The orthogonal projection matrix (see Section 2.3.3.1) for projecting onto V is

$$\mathbf{P}_V = \mathbf{V}\mathbf{V}^\top$$

as $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$.
The coordinates of the data points projected onto V (with respect to the basis for V) are the **principal component scores**:

$$\mathbf{y}_i = \begin{pmatrix} y_{i1} \\ \vdots \\ y_{ir} \end{pmatrix} = \mathbf{V}^\top \mathbf{x}_i$$

where

$$\mathbf{V} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \dots & \mathbf{v}_r \\ | & & | \end{pmatrix}$$

is the matrix of right singular vectors from the SVD of $\mathbf{X}$. The transformed variables are

$$\mathbf{Y} = \begin{pmatrix} - & \mathbf{y}_1^\top & - \\ - & .. & - \\ - & \mathbf{y}_n^\top & - \end{pmatrix} = \mathbf{X}\mathbf{V}.$$

Substituting the SVD for $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ we can see the transformed variable matrix/principal component scores are

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}.$$

$\mathbf{Y}$ is a $n \times r$ matrix, and so if $r < p$ we have reduced the dimension of $\mathbf{X}$, keeping the most important parts of the data

4.1.5 Example

We consider the marks of $n = 10$ students who studied G11PRB and G11STA.

student	PRB	STA
1	81	75
2	79	73
3	66	79
4	53	55
5	43	53
6	59	49
7	62	72
8	79	92
9	49	58
10	55	56

These data haven't been column centered, so let's do that in R. You can do it using the centering matrix as previously, but here is a different approach:

```
secondyr <- data.frame(
  student = 1:10,
PRB=c(81 , 79 , 66 , 53 , 43 , 59 , 62 , 79 , 49 , 55),
STA =c(75 , 73 , 79 , 55 , 53 , 49 , 72 , 92 , 58 , 56)
        )
xbar <- colMeans(secondyr[,2:3]) #only columns 2 and 3 are data
X <- as.matrix(sweep(secondyr[,2:3], 2, xbar) )
```

PRB	STA
18.4	8.8
16.4	6.8
3.4	12.8
-9.6	-11.2
-19.6	-13.2
-3.6	-17.2
-0.6	5.8
16.4	25.8
-13.6	-8.2
-7.6	-10.2

The sample covariance matrix can be computed in two ways:

```
1/10* t(X)%*%X
```

```
##         PRB    STA
## PRB 162.04 135.38
## STA 135.38 175.36
```

```
cov(X)*9/10
```

```
##         PRB    STA
## PRB 162.04 135.38
## STA 135.38 175.36
```

```
# Remember R uses the unbiased factor 1/(n-1),
# so the 9/10=(n-1)/n changes this to 1/n
# to match the notes
```

We can find the singular value decomposition of $\mathbf{X}$ using R

```
(X_svd = svd(X))
```

```
## $d
## [1] 55.15829 18.20887
```

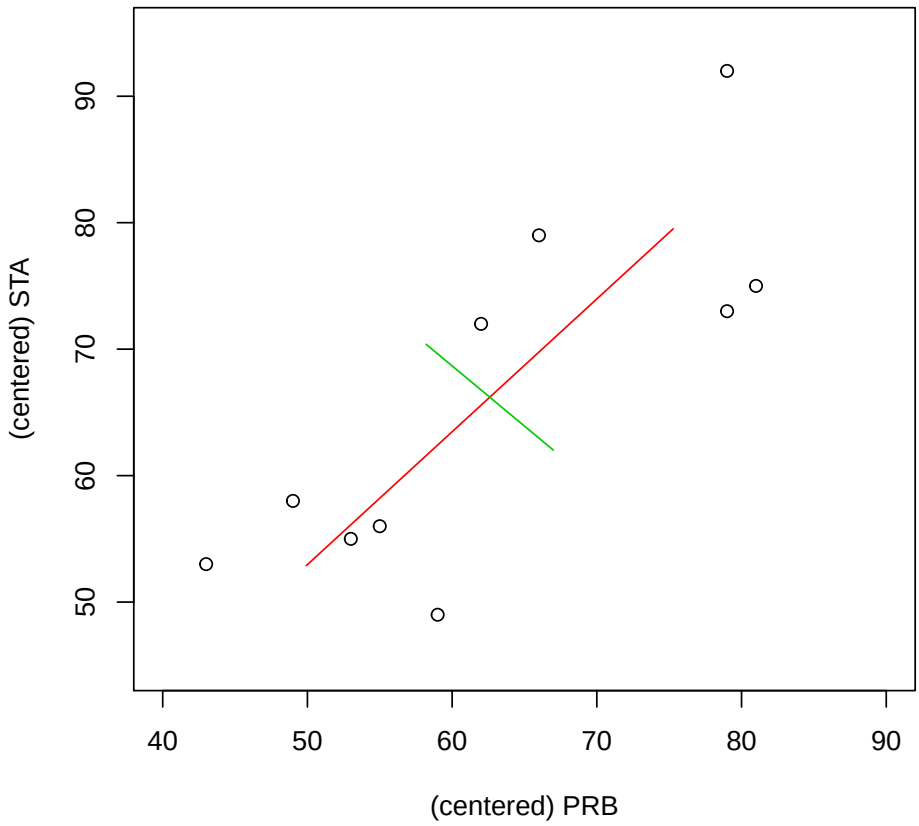
```
##
## $u
##               [,1]        [,2]
##  [1,] -0.34556317 -0.39864295
##  [2,] -0.29430029 -0.39482564
##  [3,] -0.21057607  0.34946080
##  [4,]  0.26707104 -0.04226416
##  [5,]  0.41833934  0.27975879
##  [6,]  0.27085156 -0.50812066
##  [7,] -0.06865802  0.24349429
##  [8,] -0.54378479  0.32464825
##  [9,]  0.27768146  0.23043980
## [10,]  0.22893893 -0.08394852
##
## $v
##            [,1]       [,2]
## [1,] -0.6895160 -0.7242705
## [2,] -0.7242705  0.6895160
```

So we can see that the eigenvectors/right singular vectors/loadings are

$$\mathbf{v}_1 = \begin{pmatrix} -0.69 \\ -0.724 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} -0.724 \\ 0.69 \end{pmatrix}$$

Sometimes the new variables have an obvious interpretation. In this case the first PC gives approximately equal weight to PRB and STA and thus represents some form of negative ''average" mark. Note that the singular vectors are only determined upto multiplication by ± 1. In this case, R has chosen $\mathbf{v}_1$ to have negative entries, but we could multiply $\mathbf{v}_1$ by -1 so that the first PC was more like the avearge. As it is, a student that has a high mark on PRB and STA will have a low negative value for y_1. The second PC, meanwhile, represents a contrast between PRB and STA. For example, a large positive value for y_2 implies the student did much better on STA than PRB, and a large negative value implies the opposite.

If we plot the data along with the principal components. The two lines, centred on $\bar{\mathbf{x}}$, are in the direction of the principal components/eigenvectors, and their lengths are $2\sqrt{\lambda_j}$, $j = 1, 2$. We can see that the first PC is in the direction of greatest variation (shown in red), and that the second PC (shown in green) is orthogonal to the first PC.



We can find the transformed variables by computing either $\mathbf{XV}$ or $\mathbf{U\Sigma}$

```
X %*% X_svd$v
```

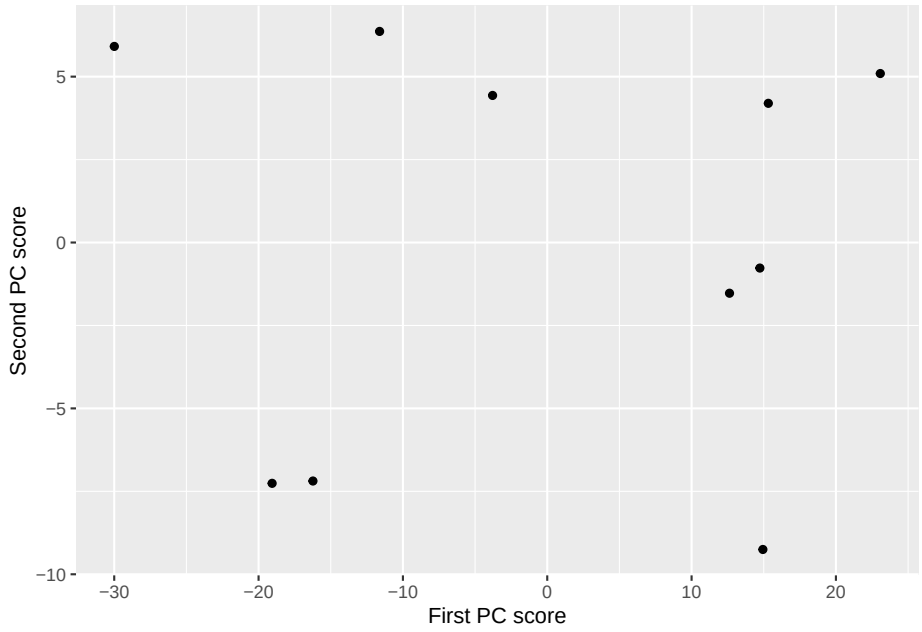
```
##              [,1]       [,2]
##  [1,] -19.060674 -7.2588361
##  [2,] -16.233101 -7.1893271
##  [3,] -11.615016  6.3632849
##  [4,]  14.731183 -0.7695824
##  [5,]  23.074883  5.0940904
##  [6,]  14.939710 -9.2523011
##  [7,]  -3.787059  4.4337549
##  [8,] -29.994240  5.9114764
##  [9,]  15.316435  4.1960474
## [10,]  12.627880 -1.5286074
```

```
X_svd$u %*% diag(X_svd$d)
```

```
##              [,1]       [,2]
##  [1,] -19.060674 -7.2588361
##  [2,] -16.233101 -7.1893271
```

```
##  [3,] -11.615016  6.3632849
##  [4,]  14.731183 -0.7695824
##  [5,]  23.074883  5.0940904
##  [6,]  14.939710 -9.2523011
##  [7,]  -3.787059  4.4337549
##  [8,] -29.994240  5.9114764
##  [9,]  15.316435  4.1960474
## [10,]  12.627880 -1.5286074
```

If we plot the PC scores we can see that the variation is now in line with the new coordinate axes:



R also has a built-in function for doing PCA.

```
pca <- prcomp(secondyr[,2:3]) # prcomp will automatocally remove the column mean
pca$rotation # the loadings
```

```
##            PC1        PC2
## PRB -0.6895160 -0.7242705
## STA -0.7242705  0.6895160
```

```
pca$x # the scores
```

```
##              PC1        PC2
##  [1,] -19.060674 -7.2588361
##  [2,] -16.233101 -7.1893271
##  [3,] -11.615016  6.3632849
##  [4,]  14.731183 -0.7695824
```

```
##  [5,]  23.074883  5.0940904
##  [6,]  14.939710 -9.2523011
##  [7,]  -3.787059  4.4337549
##  [8,] -29.994240  5.9114764
##  [9,]  15.316435  4.1960474
## [10,]  12.627880 -1.5286074
```

Note that the new variables have sample mean $\bar{\mathbf{y}} = \mathbf{0}$. The sample covariance matrix is a diagonal with entries given by the eigenvalues (see part 4. of Proposition 4.2). Note that there is always some numerical error (so quantities are never 0, and instead are just very small numnbers).

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2) = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

```
colMeans(pca$x)
```

```
##           PC1           PC2
##  2.842171e-15 -9.769963e-16
```

```
cov(pca$x)*9/10 # to convert to using 1/n as the denominator
```

```
##               PC1          PC2
## PC1 3.042437e+02 1.974167e-14
## PC2 1.974167e-14 3.315628e+01
```

Finally, note that we did the singular value decomposition for $\mathbf{X}$ above not $\frac{1}{\sqrt{10}}\mathbf{X}$, and so we'd need to square and scale the singular values to find the eigenvalues. Let's check:

```
X_svd$d^2/10 # square and scale the singular values
```

```
## [1] 304.24372  33.15628
```

```
eigen(t(X) %*% X/10)$values  # compute the eigenvalues of the covariance matrix
```

```
## [1] 304.24372  33.15628
```

```
svd(X/sqrt(10))$d^2 # compute the singular values of X/sqrt(10) and square
```

```
## [1] 304.24372  33.15628
```

4.1.6 Example: Iris

In general when using R to do PCA, we don't need to compute the SVD and then do the projections, as there is an R command `prcomp` that will do it all for us. The `princomp` will also do PCA, but is less stable than `prcomp`, and it is recommended that you use `prcomp` in preference.

Let's do PCA on the iris dataset discussed in Chapter 1. The `prcomp` returns the square root of the eigenvalues (the standard devaiation of the PC scores), and the PC scores.

```
iris.pca = prcomp(iris[,1:4])
iris.pca$sdev # the square root of the eigenvalues
```

```
## [1] 2.0562689 0.4926162 0.2796596 0.1543862
```

```
head(iris.pca$x)  #the PC scores
```

```
##            PC1        PC2         PC3          PC4
## [1,] -2.684126 -0.3193972  0.02791483  0.002262437
## [2,] -2.714142  0.1770012  0.21046427  0.099026550
## [3,] -2.888991  0.1449494 -0.01790026  0.019968390
## [4,] -2.745343  0.3182990 -0.03155937 -0.075575817
## [5,] -2.728717 -0.3267545 -0.09007924 -0.061258593
## [6,] -2.280860 -0.7413304 -0.16867766 -0.024200858
```

The PC loadings/eigenvectors can also be accessed, as can the sample mean

```
iris.pca$rotation #the eigenvecstors
```

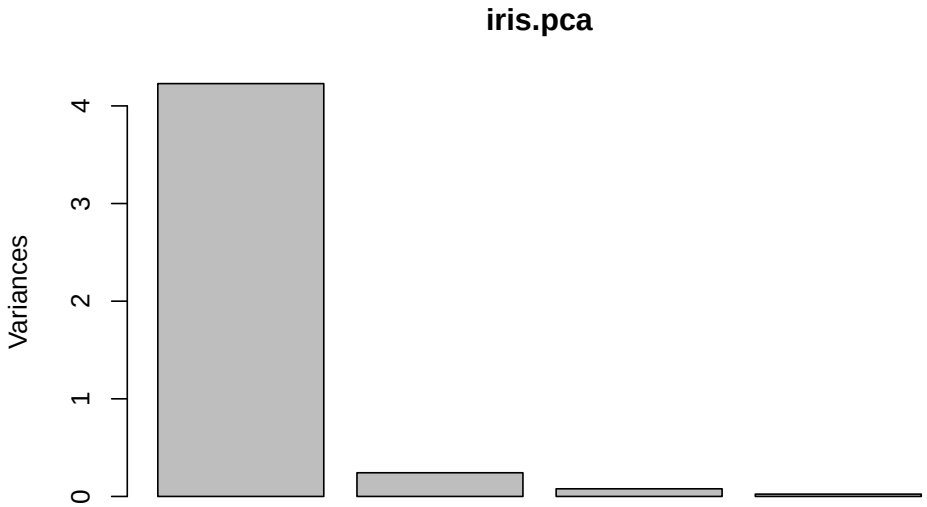
```
##                     PC1         PC2         PC3        PC4
## Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

```
iris.pca$center # the sample mean of the data
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

A scree plot can be obtained simply by using the `plot` command. The summary command also gives useful information about the importance of each PC.

```
plot(iris.pca)
```

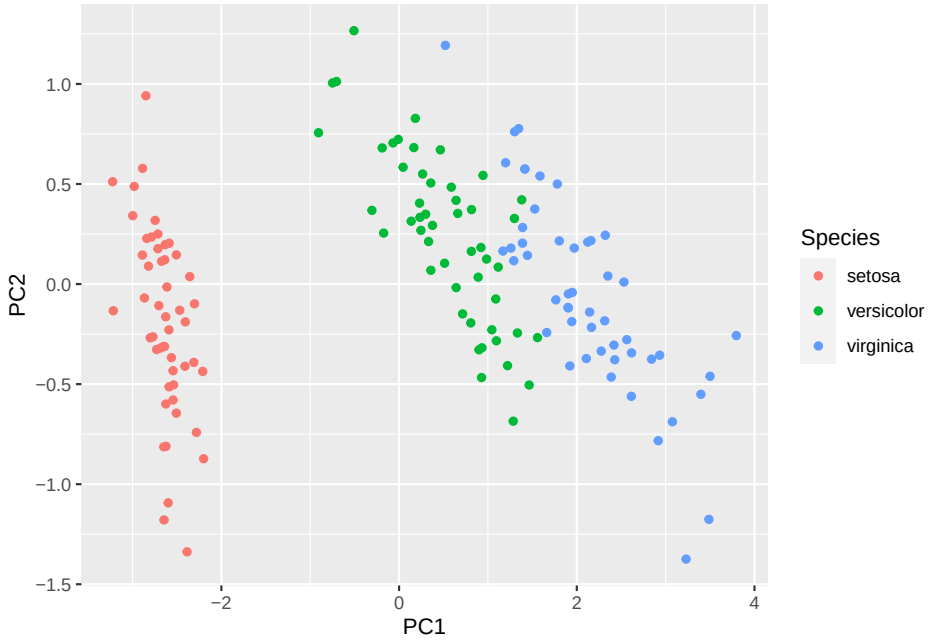



```
summary(iris.pca)
```

```
## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

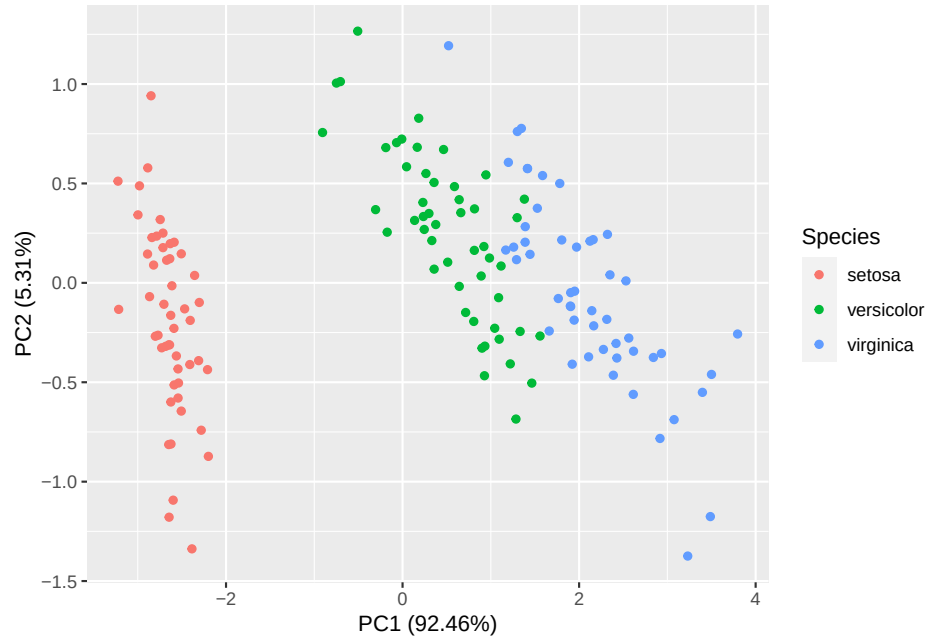
To plot the PC scores, you can either manually create a plot or use the `ggfortify` package. For example, here is a plot of the first two PC scores coloured according to the species of iris.

```
iris$PC1=iris.pca$x[,1]
iris$PC2=iris.pca$x[,2]
qplot(PC1, PC2, colour=Species, data=iris)
```



The `ggfortify` package provides a nice wrapper for some of this functionality.

```
library(ggfortify)
autoplot(iris.pca, data = iris, colour = 'Species', scale=FALSE)
```



4.2 PCA: a formal description with proofs

Let's now summarize what we've said so far and prove some results about principal component analysis.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ denote a sample of vectors in $\mathbb{R}^p$ with sample mean vector $\bar{\mathbf{x}}$ and sample covariance matrix $\mathbf{S}$. Suppose $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top \mathbf{H}\mathbf{X}$ has spectral decomposition (see Proposition 3.3)

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top = \sum_{j=1}^{p} \lambda_j \mathbf{v}_j \mathbf{v}_j^\top, \tag{4.3}$$

where the eigenvalues are $\lambda_1 \geq \lambda_2 \geq \lambda_p \geq 0$ with $\mathbf{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_p\}$, and $\mathbf{V}$ contains the eigenvectors of $\mathbf{S}$. If $\mathbf{S}$ is of full rank, then $\lambda_p > 0$. If $\mathbf{S}$ is rank r, with $r < p$, then $\lambda_{r+1} = \ldots, \lambda_p = 0$ and we can truncate $\mathbf{V}$ to consider just the first r columns.

The principal components of $\mathbf{X}$ are defined sequentially. If the $\mathbf{v}_k$ is value of $\mathbf{u}$ that maximizes the objective for the k^{th} problem (for $k < j$), then the j^{th} principal component is the solution to the following optimization problem:

$$\max_{\mathbf{u}:\, ||\mathbf{u}||=1} \mathbf{u}^\top \mathbf{S}\mathbf{u} \tag{4.4}$$

subject to

$$\mathbf{v}_k^\top \mathbf{u} = 0, \qquad k = 1, \ldots, j-1. \tag{4.5}$$

(for $j = 1$ there is no orthogonality constraint).

Proposition 4.1. *The maximum of Equation* (4.4) *subject to Equation* (4.5) *is equal to* λ_j *and is obtained when* $\mathbf{u} = \mathbf{v}_j$.

Proof. We can prove this using the method of Lagrange multipliers. For $j = 1$ our objective is

$$\mathcal{L} = \mathbf{u}^\top \mathbf{S}\mathbf{u} + \lambda(1 - \mathbf{u}^\top \mathbf{u})$$

Differentiating (see 2.1.4) with respect to $\mathbf{u}$ and setting the derivative equal to zero gives

$$2\mathbf{S}\mathbf{u} - 2\lambda\mathbf{u} = 0$$

Rearranging we see that $\mathbf{u}$ must satify

$$\mathbf{S}\mathbf{u} = \lambda\mathbf{u} \text{ with } \mathbf{u}^\top \mathbf{u} = 1$$

i.e., $\mathbf{u}$ is a unit eigenvector of $\mathbf{S}$. Substituting this back in to the objective we see

$$\mathbf{u}\mathbf{S}\mathbf{u} = \lambda$$

and so we must choose $\mathbf{u} = \mathbf{v}_1$, the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}$.

We now proceed inductively and assume the result is true for $k = 1, \ldots, j-1$. The Lagrangian for the j^{th} optimization problem is

$$\mathcal{L} = \mathbf{u}^\top \mathbf{S}\mathbf{u} + \lambda(1 - \mathbf{u}^\top \mathbf{u}) + \sum_{k=1}^{j-1} \mu_k (0 - \mathbf{u}^\top \mathbf{v}_k)$$

where we now have j Lagrange multipliers $\lambda, \mu_1, \ldots, \mu_{j-1}$ - one for each constraint. Differentiating with respect to $\mathbf{u}$ and setting equal to zero gives

$$0 = 2\mathbf{S}\mathbf{u} - 2\lambda\mathbf{u} - \sum_{k=1}^{j-1} \mu_k \mathbf{v}_k = 0$$

If we left multiply by $\mathbf{v}_l^\top$ we get

$$2\mathbf{v}_l^\top \mathbf{S}\mathbf{u} - 2\lambda \mathbf{v}_l \mathbf{u} - \sum \mu_k \mathbf{v}_l^\top \mathbf{v}_k = 0$$

We know $\mathbf{v}_l$ is an eigenvector of $\mathbf{S}$ and so $\mathbf{S}\mathbf{v}_l = \lambda_l \mathbf{v}_l$ and hence $\mathbf{v}_k \mathbf{S}\mathbf{u} = 0$ as $\mathbf{v}_l^\top \mathbf{u} = 0$. Also

$$\mathbf{v}_l^\top \mathbf{v}_k = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise,} \end{cases}$$

and thus we've shown that $\mu_l = 0$ for $l = 1, \ldots, j-1$. So again we have that

$$\mathbf{S}\mathbf{u} = \lambda\mathbf{u}$$

i.e., $\mathbf{u}$ must be a unit eigenvector of $\mathbf{S}$. It only remains to show *which* eigenvector it is. Because $\mathbf{u}$ must be orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}$, and as $\mathbf{v}_l^\top \mathbf{S}\mathbf{v}_l = \lambda_l$, we must choose $\mathbf{u} = \mathbf{v}_j$, the eigenvector corresponding to the j^{th} largest eigenvalue. □

4.2.1 Properties of principal components

For $j = 1, \ldots, p$, the scores of the j^{th} principal component (PC) are given by

$$y_{ij} = \mathbf{v}_j^\top (\mathbf{x}_i - \bar{\mathbf{x}}), \qquad i = 1, \ldots, n.$$

The j^{th} eigenvector $\mathbf{v}_j$ is sometimes referred to as the vector of **loadings** for the j^{th} PC. Note that if $\text{rank}(S) = r < p$, then the $r+1^{th}, \ldots, p^{th}$ scores are meaningless, as they will all be zero.

In vector notation

$$\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{ip})^\top = \mathbf{V}^\top (\mathbf{x}_i - \bar{\mathbf{x}}), \qquad i = 1, \ldots, n.$$

In matrix form, the full set of PC scores is given by

$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top = \mathbf{H}\mathbf{X}\mathbf{V}.$$

If $\tilde{\mathbf{X}} = \mathbf{HX}$ is the column centered data matrix, with singular value decomposition $\tilde{\mathbf{X}} = \mathbf{U\Sigma V}^\top$ with $\mathbf{V}$ as in Equation (4.3), then

$$\mathbf{Y} = \tilde{\mathbf{X}}\mathbf{V} = \mathbf{U\Sigma}.$$

The transformed variables $\mathbf{Y} = \mathbf{HXV}$ have some important properties which we collect together in the following proposition.

Proposition 4.2. *The following results hold:*

1. *The sample mean vector of* $\mathbf{y}_1, \ldots, \mathbf{y}_n$ *is the zero vector:* $\bar{\mathbf{y}} = \mathbf{0}_p$
2. *The sample covariance matrix of* $\mathbf{y}_1, \ldots, \mathbf{y}_n$ *is*

   $$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_p)$$

   i.e., for each fixed j*, the sample variance of* y_{ij} *is* λ_j*, and* y_{ij} *is uncorrelated with with* y_{ik} *for* $j \neq k$*.*
3. *For* $j \leq k$ *the sample variance of* $\{y_{ij}\}_{i=1,\ldots,n}$ *is greater than or equal to the sample variance of* $\{y_{ik}\}_{i=1,\ldots,n}$*.*

   $$\mathbf{v}_1^\top \mathbf{S}\mathbf{v}_1 \geq \mathbf{v}_2^\top \mathbf{S}\mathbf{v}_2 \geq \ldots \geq \mathbf{v}_p^\top \mathbf{S}\mathbf{v}_p \geq 0$$

   Note that if $\text{rank}(S) = r < p$*, then* $\mathbf{v}_k^\top \mathbf{S}\mathbf{v}_k = 0$ *for* $k = r+1, \ldots, p$*.*
4. *The sum of the sample variances is equal to the trace of* $\mathbf{S}$

   $$\sum_{j=1}^{p} \mathbf{v}_j^\top \mathbf{S}\mathbf{v}_j = \sum_{j=1}^{p} \lambda_j = tr(\mathbf{S})$$

5. *The product of the sample variances is equal to the determinant of* $\mathbf{S}$

   $$\prod_{j=1}^{p} \mathbf{v}_j^\top \mathbf{S}\mathbf{v}_j = \prod_{j=1}^{p} \lambda_j = |\mathbf{S}|.$$

Proof. For i.

$$\bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{V}^\top(\mathbf{x}_i - \bar{\mathbf{x}}) = \frac{1}{n}\mathbf{V}^\top \sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}}) = \mathbf{0}.$$

For 2. the sample covariance matrix of $\mathbf{y}_1, \ldots, \mathbf{y}_n$ is

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^\top &= \frac{1}{n}\sum \mathbf{V}^\top(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \mathbf{x})^\top \mathbf{V} \\
&= \mathbf{V}^\top \mathbf{S}\mathbf{V} \\
&= \mathbf{V}^\top \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top \mathbf{V} \text{ substiting the spectral decomposition for } \mathbf{S} \\
&= \mathbf{\Lambda}
\end{aligned}$$

3. is a consequence 2. and of ordering the eigenvalues in decreasing magnitude.
4. follows from lemma 2.1 and the spectral decomposition of $\mathbf{S}$:

$$\text{tr}(\mathbf{S}) = \text{tr}(\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top) = \text{tr}(\mathbf{V}^\top\mathbf{V}\mathbf{\Lambda}) = \text{tr}(\mathbf{\Lambda}) = \sum \lambda_i$$

5. follows from 3.2.

□

From these properties we say that a proportion

$$\frac{\lambda_j}{\lambda_1 + \ldots + \lambda_p}$$

of the variability in the sample is 'explained' by the j^{th} PC.

One tool for looking at the contributions of each PC is to look at the **scree plot** which plots the percentage of variance explained by PC j against j. We'll see examples of scree plots below.

4.2.2 Example: Football

We can apply PCA to a football league table where W, D, L are the number of matches won, drawn and lost and G and GA are the goals scored for and against, and GD is the goal difference $(G - GA)$. An extract of the table for the 2019-2020 Premier League season is:

Team	W	D	L	G	GA	GD
Liverpool	32	3	3	85	33	52
Manchester City	26	3	9	102	35	67
Manchester United	18	12	8	66	36	30
Chelsea	20	6	12	69	54	15
Leicester City	18	8	12	67	41	26
Tottenham Hotspur	16	11	11	61	47	14
Wolverhampton	15	14	9	51	40	11
Arsenal	14	14	10	56	48	8
Sheffield United	14	12	12	39	39	0
Burnley	15	9	14	43	50	-7

The sample mean vector is

$$\bar{\mathbf{x}} = \begin{pmatrix} 14.4 \\ 9.2 \\ 14.4 \\ 51.7 \\ 51.7 \\ 0 \end{pmatrix}.$$

Note that the total goals scored must equal the total goals conceded, and that the sum of the goal differences must be 0. The sample covariance matrix is

$$\mathbf{S} = \begin{pmatrix} 38.3 & -9.18 & -29.2 & 103 & -57 & 160 \\ -9.18 & 10.2 & -0.98 & -27.5 & -2.24 & -25.2 \\ -29.2 & -0.98 & 30.1 & -75.3 & 59.3 & -135 \\ 103 & -27.5 & -75.3 & 336 & -147 & 483 \\ -57 & -2.24 & 59.3 & -147 & 134 & -281 \\ 160 & -25.2 & -135 & 483 & -281 & 764 \end{pmatrix} \tag{4.6}$$

The eigenvalues of $\mathbf{S}$ are

$$\mathbf{\Lambda} = \text{diag}\begin{pmatrix} 1300 & 71.9 & 8.05 & 4.62 & -2.65e-14 & -3.73e-14 \end{pmatrix}$$

Note that we have two zero eigenvalues (which won't be computed as exactly zero because of numerical rounding errors) because two of our variables are a linear combinations of the other variables, $W + D + L = 38$ and $GD = G - GA$. The corresponding eigenvectors are

$$\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_6] = \begin{pmatrix} -0.166 & 0.0262 & -0.707 & 0.373 & 0.222 & -0.533 \\ 0.0282 & -0.275 & 0.661 & 0.391 & 0.222 & -0.533 \\ 0.138 & 0.249 & 0.0455 & -0.764 & 0.222 & -0.533 \\ -0.502 & 0.6 & 0.202 & 0.117 & 0.533 & 0.222 \\ 0.285 & 0.701 & 0.11 & 0.286 & -0.533 & -0.222 \\ -0.787 & -0.101 & 0.0915 & -0.169 & -0.533 & -0.222 \end{pmatrix}$$

The proportion of variability explained by each of the PCs is:

$$\begin{pmatrix} 0.939 & 0.052 & 0.00583 & 0.00334 & -1.92e-17 & -2.7e-17 \end{pmatrix}$$

There is no point computing the scores for PC 5 and 6, because these do not explain any of the variability in the data. Similarly, there is little value in computing the scores for PCs 3 & 4 because they account for less than 1% of the variability in the data.

We can, therefore, choose to compute only the first two PC scores. We are reducing the dimension of our data set from $p = 5$ to $p = 2$ while still retaining 99% of the variability. The first PC score/transformed variable is given by:

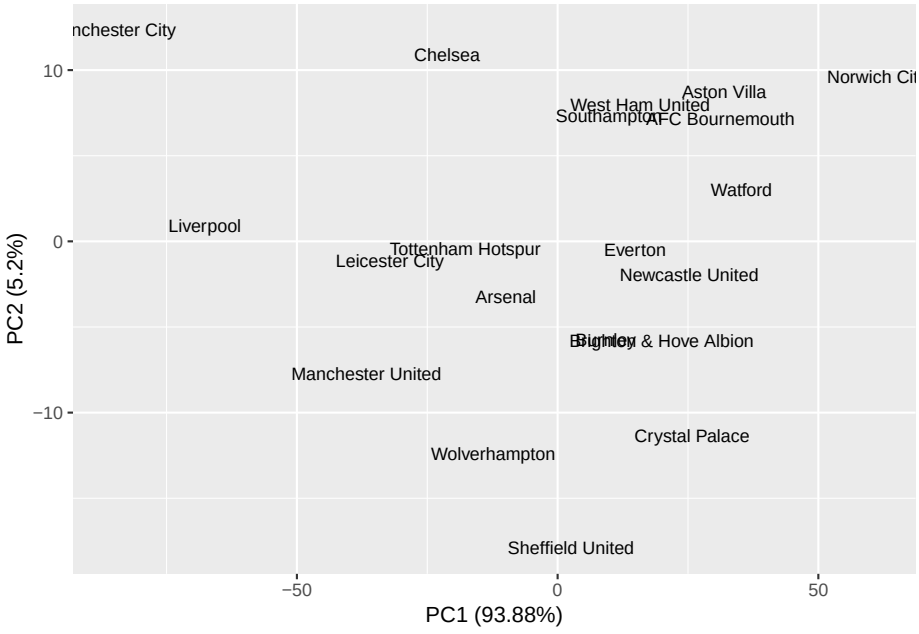
$$\begin{aligned} y_{i1} = &-0.17(W_i - \bar{W}) + 0.03(D_i - \bar{D}) + 0.14(L_i - \bar{L}) \\ &+ -0.5(G_i - \bar{G}) + 0.28(GA_i - \bar{GA}) + -0.79(GD_i - \bar{GD}), \end{aligned}$$

and similarly for PC 2.

The first five rows of our revised 'league table' are now

Team	PC1	PC2
Liverpool	-67.6	0.9
Manchester City	-85.6	12.3
Manchester United	-36.7	-7.7
Chelsea	-21.2	10.9
Leicester City	-32.2	-1.1

Now that we have reduced the dimension to $p = 2$, we can visualise the differences between the teams.



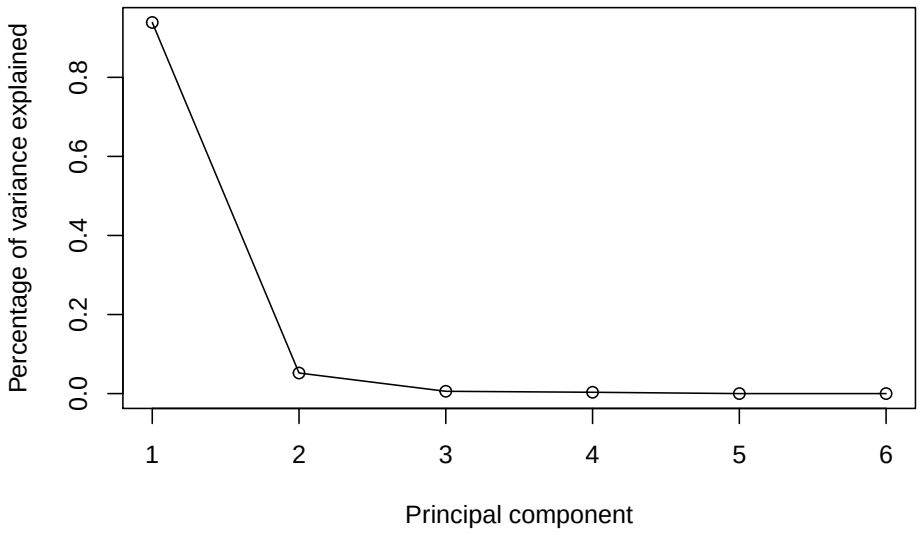
We might interpret the PCs as follows. The first PC seems to measure the difference in goals scored and conceded between teams. Low values of PC1 indicate good peformance, and high values poor performance. Teams are rewarded with -0.79 for each positive goal difference, and -0.5 for each goal scored, whilst being penalised by 0.28 for every goal they concede. So a team with a large negative PC1 score tends to score lots of goals and concede few. If we rank teams by their PC1 score, and compare this with the rankings using 3 points for a win and 1 point for a draw we get a different ranking of the teams.

	PC1	PC2
Manchester City	-85.59	12.35
Liverpool	-67.64	0.93
Manchester United	-36.66	-7.73
Leicester City	-32.16	-1.13
Chelsea	-21.19	10.90

The second PC has a strong positive loading for both goals for and against. A team with a large positive PC 2 score was, therefore, involved in matches with lots of goals. We could, therefore, interpret PC 2 as an 'entertainment' measure, ranking teams according to their involvement in high-scoring games.

The above example raises the question of how many PCs should we use in practice. If we reduce the dimension to $p = 1$ then we can rank observations and analyse our new variable with univariate statistics. If we reduce the dimension to $p = 2$ then it is still easy to visualise the data. However, reducing the dimension to $p = 1$ or $p = 2$ may involve losing lots of information and a sensible answer should depend on the objectives of the analysis and the data itself.

The scree graph for the football example is:



There are many possible methods for choosing the number of PCs to retain for analysis, including:

- retaining enough PCs to explain, say, 90% of the total variation;
- retaining PCs where the eigenvalue is above the average.

To retain enough PCs to explain 90% of the total variance, would require us to keep just a single PCs in this case.

4.2.3 PCA based on R versus PCA based on S

Recall the distinction between the sample covariance matrix **S** and the sample correlation matrix **R**. Note that all correlation matrices are also covariance matrices, but not all covariance matrices are correlation matrices. Before doing PCA we must decide whether to do PCA based on **S** or **R**? As we will see later

- PCA based on **R** (but not **S**) is scale invariant, whereas
- PCA based on **S** is invariant under orthogonal rotation.

If the original p variables represent very different types of quantity or show marked differences in variances, then it will usually be better to use **R** rather than **S**. However, in some circumstances, we may wish to use **S**, such as when the p variables are measuring similar entities and the sample variances are not too different.

Given that the required numerical calculations are easy to perform in R, we might wish to do it both ways and see if it makes much difference. To use the correlation matrix **R**, we just add the option `scale=TRUE` when using the `prcomp` command.

4.2.3.1 Football example continued

If we repeat the analysis of the football data using **R** instead of **S**, we get find principal components:

$$\mathbf{\Lambda} = \text{diag}\begin{pmatrix} 4.51 & 1.25 & 0.156 & 0.0863 & 3.68e-32 & 2.48e-33 \end{pmatrix}$$

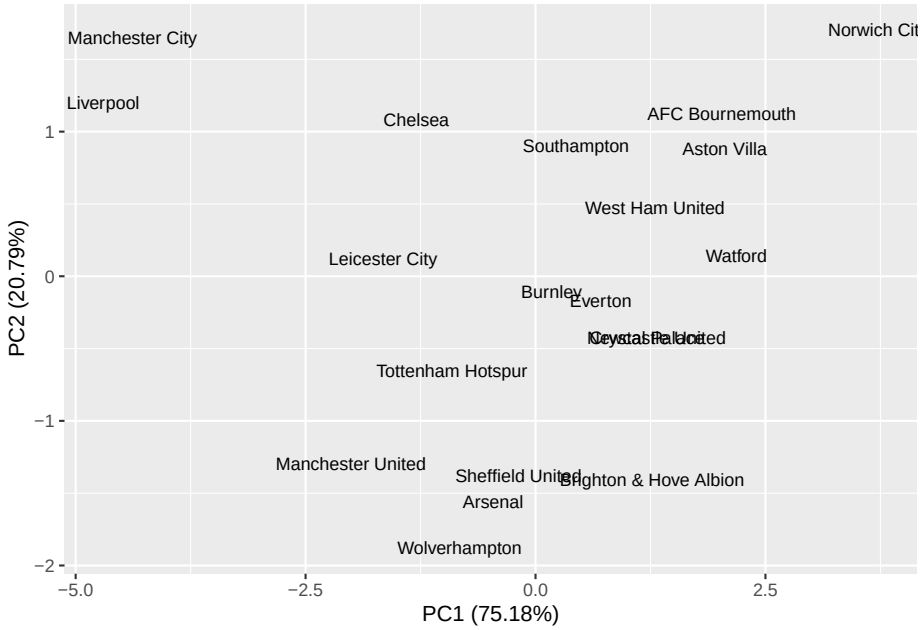
$$\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_6] = \begin{pmatrix} -0.456 & 0.149 & -0.342 & -0.406 & 0.466 & 0.52 \\ 0.143 & -0.844 & 0.344 & -0.143 & 0.24 & 0.268 \\ 0.432 & 0.321 & 0.186 & 0.541 & 0.413 & 0.461 \\ -0.438 & 0.214 & 0.7 & -0.0181 & 0.389 & -0.348 \\ 0.419 & 0.342 & 0.386 & -0.671 & -0.245 & 0.22 \\ -0.466 & -0.00136 & 0.302 & 0.269 & -0.586 & 0.525 \end{pmatrix}$$

The effect of using **R** is to standardize each of the original variables to have variance 1. The first PC now has loadings which are more evenly balanced across the 6 original variables.

Teams will have a small value of PC1 score if they won lots, lost rarely, scored a lot, and conceded rarely. In other words, PC1 is a complete measure of overall performance. If we look at the league table based on ordering according to PC1 we get a table that looks more like the original table.

	PC1	PC2
Liverpool	-4.70	1.20
Manchester City	-4.38	1.65
Manchester United	-2.01	-1.29
Chelsea	-1.29	1.08
Leicester City	-1.66	0.12
Tottenham Hotspur	-0.91	-0.65
Wolverhampton	-0.82	-1.88
Arsenal	-0.46	-1.56
Sheffield United	-0.18	-1.38
Burnley	0.18	-0.10

Overall for these data, doing PCA with $\mathbf{R}$ instead of $\mathbf{S}$ better summarizes the data (although this is just my subjective opinion - you may feel differently).



4.2.4 Population PCA

So far we have considered sample PCA based on the sample covariance matrix or sample correlation matrix:

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top.$$

We note now that there is a *population* analogue of PCA based on the population covariance matrix $\mathbf{\Sigma}$. Although the population version of PCA is not of as

much direct practical relevance as sample PCA, it is nevertheless of conceptual importance.

Let $\mathbf{x}$ denote a $p \times 1$ random vector with $\mathbb{E}(\mathbf{x}) = \boldsymbol{\mu}$ and $\mathbb{V}\mathrm{ar}(\mathbf{x}) = \boldsymbol{\Sigma}$. As defined, $\boldsymbol{\mu}$ is the population mean vector and $\boldsymbol{\Sigma}$ is the population covariance matrix.

Since $\boldsymbol{\Sigma}$ is symmetric, the spectral decomposition theorem tells us that

$$\boldsymbol{\Sigma} = \sum_{j=1}^{p} \check{\lambda}_j \check{\mathbf{v}}_j \check{\mathbf{v}}_j^\top = \check{\mathbf{V}} \check{\boldsymbol{\Lambda}} \check{\mathbf{V}}^\top$$

where the 'check' symbol $\check{}$ is used to distinguish population quantities from their sample analogues.

Then:

- the first population PC is defined by $Y_1 = \check{\mathbf{v}}_1^\top (\mathbf{x} - \boldsymbol{\mu})$;
- the second population PC is defined by $Y_2 = \check{\mathbf{v}}_2^\top (\mathbf{x} - \boldsymbol{\mu})$;
- ...
- the pth population PC is defined by $Y_p = \check{\mathbf{v}}_p^\top (\mathbf{x} - \boldsymbol{\mu})$.

The $Y_1, \ldots, Y_p$ are random variables, unlike the sample PCA case, where the y_{ij} are observed quantities. In the sample PCA case, the y_{ij} can often be regarded as the observed values of random variables.

In matrix form, the above definitions can be summarised by writing

$$\mathbf{y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \ldots \\ \ldots \\ Y_p \end{pmatrix} = \check{\mathbf{V}}^\top (\mathbf{x} - \boldsymbol{\mu}).$$

The population PCA analogues of the sample PCA properties listed in Proposition 4.2 are now given. Note that the Y_j's are random variables as opposed to observed values of random variables.

Proposition 4.3. *The following results hold for the random variables $Y_1, \ldots, Y_p$ defined above.*

1. $\mathbb{E}(Y_j) = 0$ *for* $j = 1, \ldots, p$;
2. $\mathbb{V}\mathrm{ar}(Y_j) = \check{\lambda}_j$ *for* $j = 1, \ldots, p$;
3. $\mathbb{C}\mathrm{ov}(Y_j, Y_k) = 0$ *if* $j \neq k$;
4. $\mathbb{V}\mathrm{ar}(Y_1) \geq \mathbb{V}\mathrm{ar}(Y_2) \geq \cdots \geq \mathbb{V}\mathrm{ar}(Y_p) \geq 0$;
5. $\sum_{j=1}^{p} \mathbb{V}\mathrm{ar}(Y_j) = \sum_{j=1}^{p} \check{\lambda}_j = tr(\boldsymbol{\Sigma})$;
6. $\prod_{j=1}^{p} Var(Y_j) = \prod_{j=1}^{p} \check{\lambda}_j = |\boldsymbol{\Sigma}|$.

Note that, defining $\mathbf{y} = (Y_1, \ldots, Y_p)^\top$ as before, part 1. implies that $\mathbb{E}(\mathbf{y}) = \mathbf{0}_p$ and parts 2. and 3. together imply that

$$\mathrm{Var}(\mathbf{y}) = \boldsymbol{\Lambda} \equiv \mathrm{diag}(\check{\lambda}_1, \ldots, \check{\lambda}_p).$$

Consider now a repeated sampling framework in which we assume that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are IID random vectors from a population with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

What is the relationship between the sample PCA based on the sample of observed vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and the population PCA based on the unobserved random vector $\mathbf{x}$, from the same population?

If the elements of $\boldsymbol{\Sigma}$ are all finite, then as n increases, the elements of the sample covariance matrix $\mathbf{S}$ will converge to the corresponding elements of the population covariance matrix $\boldsymbol{\Sigma}$. Consequently, we expect the principal components from sample PCA to converge to the population PCA values as n grows large. Justification of this statement comes from the weak law of large numbers applied to the components of Σ, but the details are beyond the scope of this module.

4.2.5 PCA under transformations of variables

We'll now consider what happens to PCA when the data are transformed in various ways.

Addition transformation

Firstly, consider the transformation of addition where, for example, we add a fixed amount to each variable. We can write this transformation as $\mathbf{z}_i = \mathbf{x}_i + \mathbf{c}$, where $\mathbf{c}$ is a fixed vector. Under this transformation the sample mean changes, $\bar{\mathbf{z}} = \bar{\mathbf{x}} + \mathbf{c}$, but the sample variance remains $\mathbf{S}$. Consequently, the eigenvalues and eigenvectors remain the same and, therefore, so do the principal component scores/transformed vasriables,

$$\mathbf{y}_i = \mathbf{V}^\top(\mathbf{z}_i - \bar{\mathbf{z}}) = \mathbf{V}^\top(\mathbf{x}_i + \mathbf{c} - (\bar{\mathbf{x}} + \mathbf{c})) = \mathbf{V}^\top(\mathbf{x}_i - \bar{\mathbf{x}}).$$

We say that the principal components are **invariant** under the addition transformation. An important special case is to choose $\mathbf{c} = -\bar{\mathbf{x}}$ so that the PC scores are simply $\mathbf{y}_i = \mathbf{V}^\top \mathbf{z}_i$.

Scale transformation

Secondly, we consider the scale transformation where each variable is multiplied by a fixed amount. A scale transformation occurs more naturally when we convert units of measurement from, say, metres to kilometres. We can write this transformation as $\mathbf{z}_i = \mathbf{D}\mathbf{x}_i$, where $\mathbf{D}$ is a diagonal matrix with positive elements. Under this transformation the sample mean changes from $\bar{\mathbf{x}}$ to $\bar{\mathbf{z}} = \mathbf{D}\bar{\mathbf{x}}$, and the sample covariance matrix changes from $\mathbf{S}$ to $\mathbf{DSD}$. Consequently, the principal components also change.

This lack of scale-invariance is undesirable. For example, if we analysed data that included some information on distances, we don't want the answer to depend upon whether we use km, metres, or miles as the measure of distance. One solution is to scale the data using

$$\mathbf{D} = \text{diag}(s_{11}^{-1/2}, \ldots, s_{pp}^{-1/2}),$$

where s_{ii} is the ith diagonal element of $\mathbf{S}$. In effect, we have standardised all the new variables to have variance 1. In this case the sample covariance matrix of the $\mathbf{z}_i$'s is simply the sample correlation matrix $\mathbf{R}$ of the original variables, $\mathbf{x}_i$. Therefore, we can carry out PCA on the sample correlation matrix, $\mathbf{R}$, which is invariant to changes of scale.

In summary: $\mathbf{R}$ is scale-invariant while $\mathbf{S}$ is not. To do PCA on $\mathbf{R}$ in R we use the option `scale=TRUE` in the `prcomp` command.

We saw an example of this in section 4.2.3 with the football data. Because the sample variances of G and GA are much larger than the sample variances of W, D and L, doing PCA with $\mathbf{R}$ instead of $\mathbf{S}$ completely changed the analysis.

Orthogonal transformations

Thirdly, we consider a transformation by an orthogonal matrix, $\overset{p\times p}{\mathbf{A}}$, such that $\mathbf{A}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{A} = \mathbf{I}_p$, and write $\mathbf{z}_i = \mathbf{A}\mathbf{x}_i$. This is equivalent to rotating and/or reflecting the original data.

Let $\mathbf{S}$ be the sample covariance matrix of the $\mathbf{x}_i$ and let $\mathbf{T}$ be the sample covariance matrix of the $\mathbf{z}_i$. Under this transformation the sample mean changes from $\bar{\mathbf{x}}$ to $\bar{\mathbf{z}} = \mathbf{A}\bar{\mathbf{x}}$, and the sample covariance matrix $\mathbf{S}$ changes from $\mathbf{S}$ to $\mathbf{T} = \mathbf{A}\mathbf{S}\mathbf{A}^\top$.

However, if we write $\mathbf{S}$ in terms of its spectral decomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, then $\mathbf{T} = \mathbf{A}\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{A}^\top = \mathbf{B}\mathbf{\Lambda}\mathbf{B}^\top$ where $\mathbf{B} = \mathbf{A}\mathbf{V}$ is also orthogonal. It is therefore apparent that the eigenvalues of $\mathbf{T}$ are the same as those of $\mathbf{S}$; and the eigenvectors of $\mathbf{T}$ are given by $\mathbf{b}_j$ where $\mathbf{b}_j = \mathbf{A}\mathbf{v}_j$, $j = 1, \ldots, p$. The PC scores of the rotated variables are

$$\mathbf{y}_i = \mathbf{B}^\top(\mathbf{z}_i - \bar{\mathbf{z}}) = \mathbf{V}^\top\mathbf{A}^\top\mathbf{A}(\mathbf{x}_i - \bar{\mathbf{x}}) = \mathbf{V}_1^\top(\mathbf{x}_i - \bar{\mathbf{x}}),$$

and so they are identical to the PC scores of the original variables.

Therefore, under an orthogonal transformation the eigenvalues and PC scores are unchanged; the PCs are orthogonal transformations of the original PCs. We say that the principal components are **equivariant** with respect to orthogonal transformations.

4.3 An alternative view of PCA

In this section, we will again consider the situation in which the sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ have zero mean (replace $\mathbf{x}_i$ by $\mathbf{x}_i - \bar{\mathbf{x}}$ if the mean is not

zero).

To recap, in PCA to find the r leading principal components, we solve the optimization problem

$$\begin{aligned} &\text{For } k = 1, \ldots, r \text{ maximize } \mathbf{u}_k^\top \mathbf{S}\mathbf{u}_k \\ &\qquad \text{subject to } \mathbf{u}_k^\top \mathbf{u}_j = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

We can write this in the form given in the introduction to this chapter (Equation (4.1)) as

$$\begin{aligned} &\text{Maximize } \operatorname{tr}(\mathbf{U}^\top \mathbf{S}\mathbf{U}) \\ &\text{subject to } \mathbf{U}^\top \mathbf{U} = \mathbf{I}_r, \end{aligned}$$

as $\operatorname{tr}(\mathbf{U}^\top \mathbf{S}\mathbf{U}) = \sum_{k=1}^r \mathbf{u}_k^\top \mathbf{S}\mathbf{u}_k$ if $\mathbf{U}$ has columns $\mathbf{u}_1, \ldots, \mathbf{u}_r$.

An equivalent problem

There is another optimization problem that we sometimes wish to solve, that turns out to be equivalent to the above, thus providing another reason why PCA is so widely used.

Suppose we want to find the best rank-r linear approximation to the data matrix $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \end{pmatrix}^\top$ (remember that we're assuming the data have been column centered, if not, replace $\mathbf{X}$ by $\mathbf{H}\mathbf{X}$). One way to think about this is seek a $p \times r$ matrix $\mathbf{U}$ for which the rank r linear model

$$f(\mathbf{y}) = \mathbf{U}\mathbf{y}$$

can be used to represent the data.

Let's choose $\mathbf{y}_i \in \mathbb{R}^r$ and $\mathbf{U}$ to minimize the sum of squared errors

$$\sum_{i=1}^n ||\mathbf{x}_i - \mathbf{U}\mathbf{y}_i||_2^2.$$

If we write

$$\mathbf{Y}^\top = \begin{pmatrix} | & & | \\ \mathbf{y}_1 & \ldots & \mathbf{y}_n \\ | & & | \end{pmatrix}$$

then

$$\begin{aligned} \sum_{i=1}^n ||\mathbf{x}_i - \mathbf{U}\mathbf{y}_i||_2^2 &= \operatorname{tr}((\mathbf{X}^\top - \mathbf{U}\mathbf{Y}^\top)^\top (\mathbf{X}^\top - \mathbf{U}\mathbf{Y}^\top)) \\ &= ||\mathbf{X}^\top - \mathbf{U}\mathbf{Y}^\top||_F^2 \end{aligned}$$

i.e., we're looking for the rank-r matrix $\mathbf{X}_r$ that minimizes $||\mathbf{X} - \mathbf{X}_r||_F = ||\mathbf{X}^\top - \mathbf{X}_r^\top||_F$, noting that we can write an arbitrary rank-r matrix as $\mathbf{X}_r^\top = \mathbf{U}\mathbf{Y}^\top$ for some $p \times r$ matrix $\mathbf{U}$ and a $n \times r$ matrix $\mathbf{Y}$.

It makes sense to restrict the columns of $\mathbf{U}$ to be orthonormal so that $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_r$ as non-orthonormal coordinates systems are confusing. We know that the $\mathbf{u} \in \mathcal{C}(\mathbf{U})$ (where $\mathcal{C}(\mathbf{U})$ is the column space of $\mathbf{U}$) that minimizes

$$||\mathbf{x} - \mathbf{u}||_2$$

is the orthogonal projection of $\mathbf{x}$ onto $\mathcal{C}(\mathbf{U})$, which given the columns of $\mathbf{U}$ are orthonormal is $\mathbf{u} = \mathbf{U}\mathbf{U}^\top\mathbf{x}$ (see Section 2.3.3.1). So we must have $\mathbf{X}_r^\top = \mathbf{U}\mathbf{U}^\top\mathbf{X}^\top$ and $\mathbf{Y}^\top = \mathbf{U}^\top\mathbf{X}^\top$.

So it remains to find the optimal choice for $\mathbf{U}$ by minimizing

$$\begin{aligned} ||\mathbf{X}^\top - \mathbf{U}\mathbf{U}^\top\mathbf{X}^\top||_F^2 &= ||\mathbf{X} - \mathbf{X}\mathbf{U}\mathbf{U}^\top||_F^2 \\ &= \mathrm{tr}((\mathbf{X} - \mathbf{X}\mathbf{U}\mathbf{U}^\top)^\top(\mathbf{X} - \mathbf{X}\mathbf{U}\mathbf{U}^\top)) \\ &= \mathrm{tr}(\mathbf{X}^\top\mathbf{X}) - 2\,\mathrm{tr}(\mathbf{U}\mathbf{U}^\top\mathbf{X}^\top\mathbf{X}) + \mathrm{tr}(\mathbf{U}\mathbf{U}^\top\mathbf{X}^\top\mathbf{X}\mathbf{U}\mathbf{U}^\top) \\ &= \mathrm{tr}(\mathbf{X}^\top\mathbf{X}) - \mathrm{tr}(\mathbf{U}^\top\mathbf{X}^\top\mathbf{X}\mathbf{U}) \end{aligned}$$

where we've used the fact $\mathrm{tr}(\mathbf{A}\mathbf{B}) = \mathrm{tr}(\mathbf{B}\mathbf{A})$ and that $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_r$.

Minimizing the equation above with respect to $\mathbf{U}$ is equivalent to maximizing

$$\mathrm{tr}(\mathbf{U}^\top\mathbf{S}\mathbf{U})$$

which is the maximum variance objective we used to introduce PCA.

So to summarize, the optimization problem

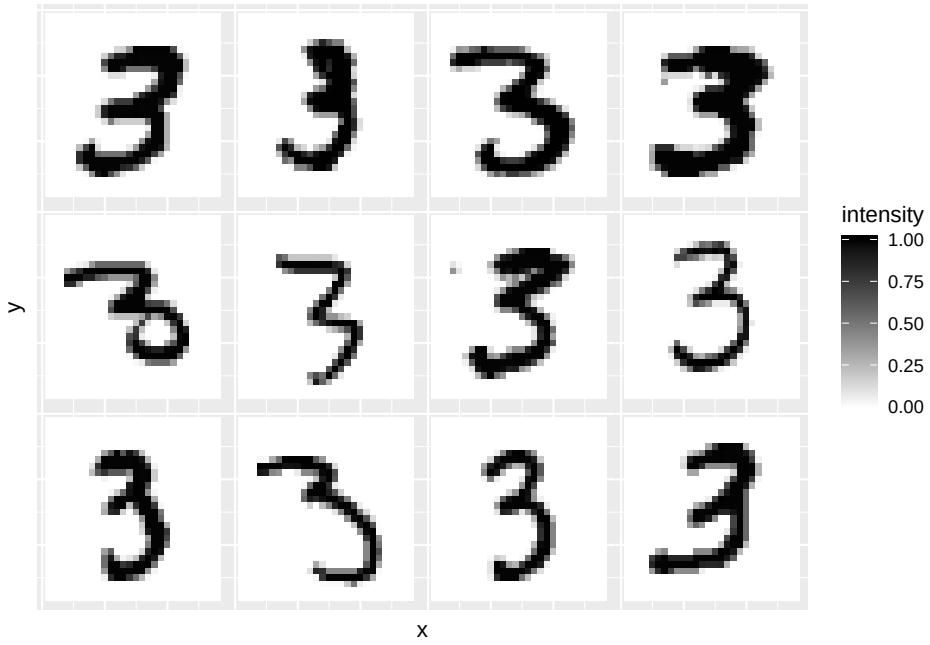
$$\begin{aligned} &\text{Minimize } ||\mathbf{X}^\top - \mathbf{U}\mathbf{U}^\top\mathbf{X}^\top||_F \\ &\text{subject to } \mathbf{U}^\top\mathbf{U} = \mathbf{I}_r, \end{aligned}$$

is equivalent to (and has the same as) the PCA optimization problem.

4.3.1 Example: MNIST handwritten digits

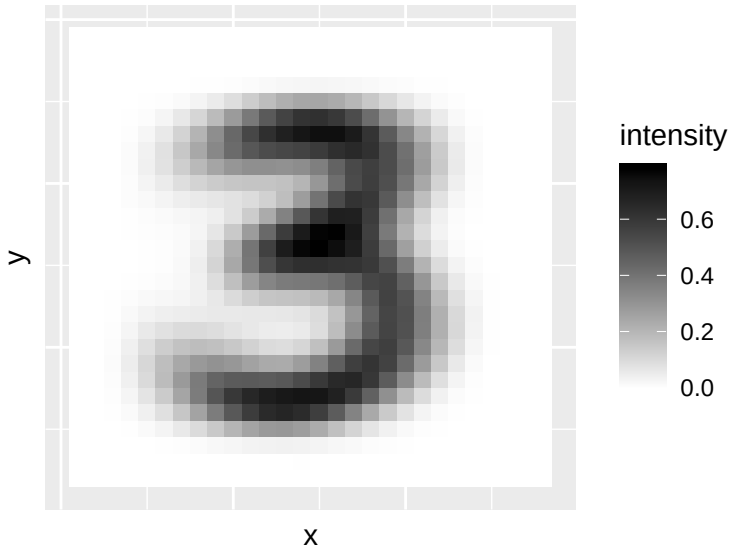
Let's consider the MNIST dataset of handwritten digits discussed in Chapter 1. Recall this is a collection of 60,000 digits, each of which has been converted to a 28×28 pixel greyscale image (so $p = 784$). I've made a clean version of the dataset available on Moodle, so you can try this analysis for yourself. Let's look at just the 3s. I've created a plotting function `plot.mnist`, which is in the code file on Moodle.

```
load(file="mnist.rda")
source('mnisttools.R')
mnist3 = mnist$train$x[mnist$train$y==3,] # select just the 3s
plot.mnist(mnist3[1:12,]) # plot the first 12 images
```



We can see there is quite a bit of variation between them. Now lets look at $\bar{\mathbf{x}}$, the average 3.

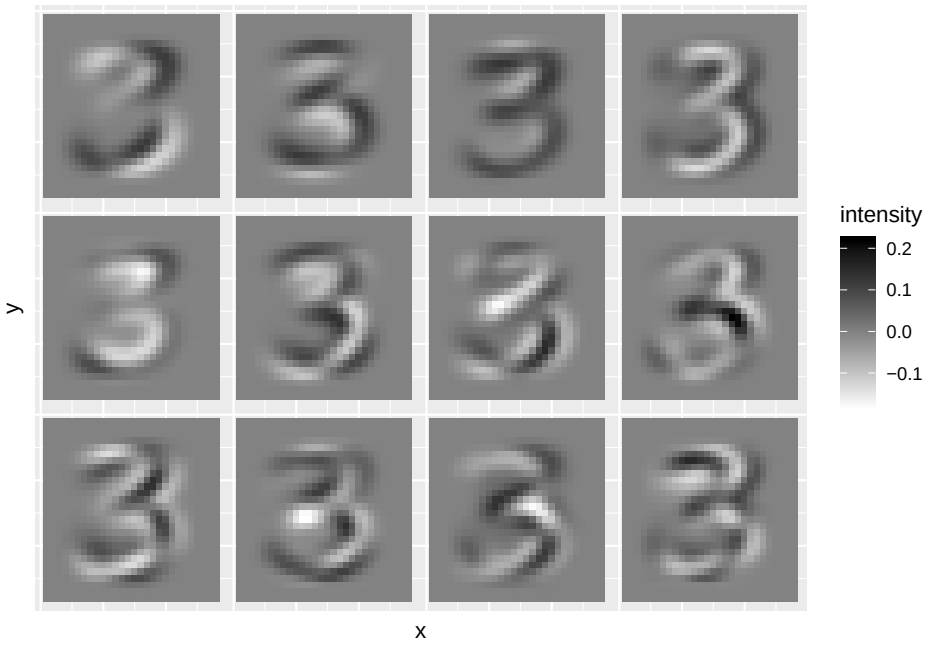
```
xbar=colMeans(mnist3)
plot.mnist(xbar)
```



We can use the `prcomp` command to find the principal components. Note that we can't use the `scale=TRUE` option as some of the columns are all 0, and so R

throws an error as it cannot rescale these to have variance 1. Let's plot the first few principal components/eigenvectors/loading vectors.

```
mnist3.pca <- prcomp(mnist3)
plot.mnist(mnist3.pca$rotation[,1:12])
```

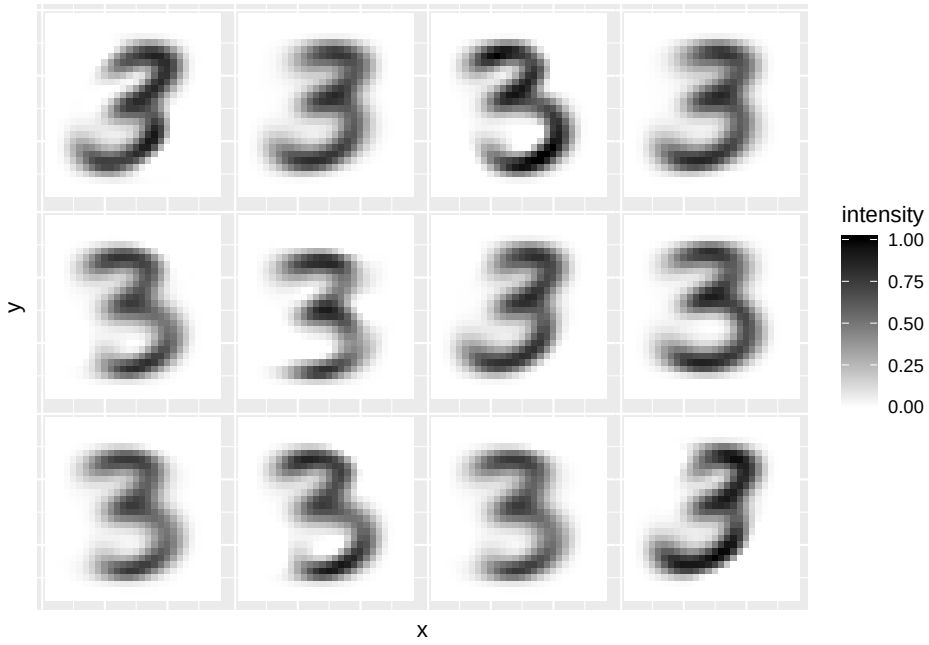


These show the main mode of variability in the 3s. Focusing on the first PC, we can see that this is a form of rotation and causes the 3 to slant either forward or backward. If we wanted a rank-2 approximation to the data we would use

$$f(\mathbf{y}) = \bar{\mathbf{x}} + y_1\mathbf{v}_1 + y_2\mathbf{v}_2$$

Let's try reconstructing the data with $r = 2$.

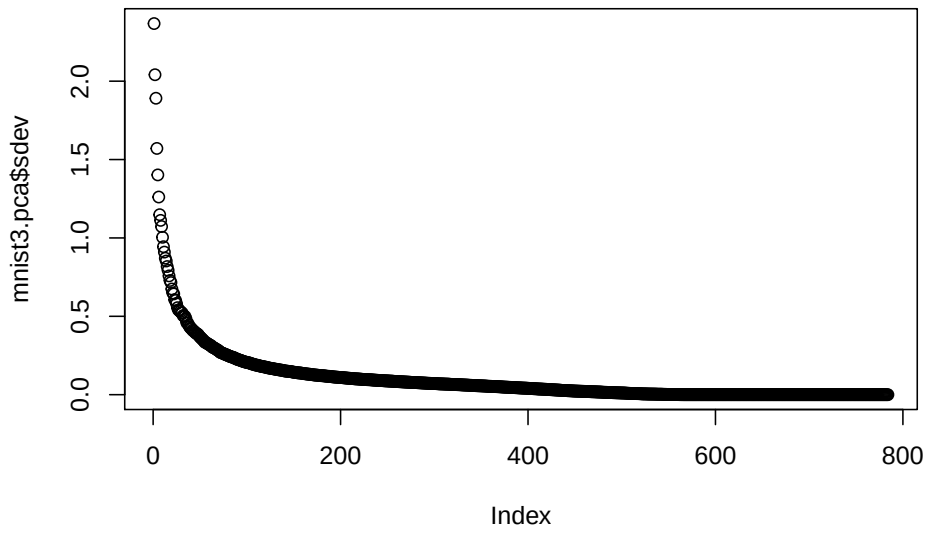
```
r=2
recon =  mnist3.pca$x[,1:r] %*% t(mnist3.pca$rotation[,1:r])
plot.mnist2(matrix(rep(xbar,12), byrow=T, nr=12)+recon[1:12,])
```



We can see that all of these 3s still look a lot like the average 3, but that they vary in their slant, and the heaviness of the line.

The scree plot shows a sharp decrease in the eigenvalues until about the 100th component, at which point they level off.

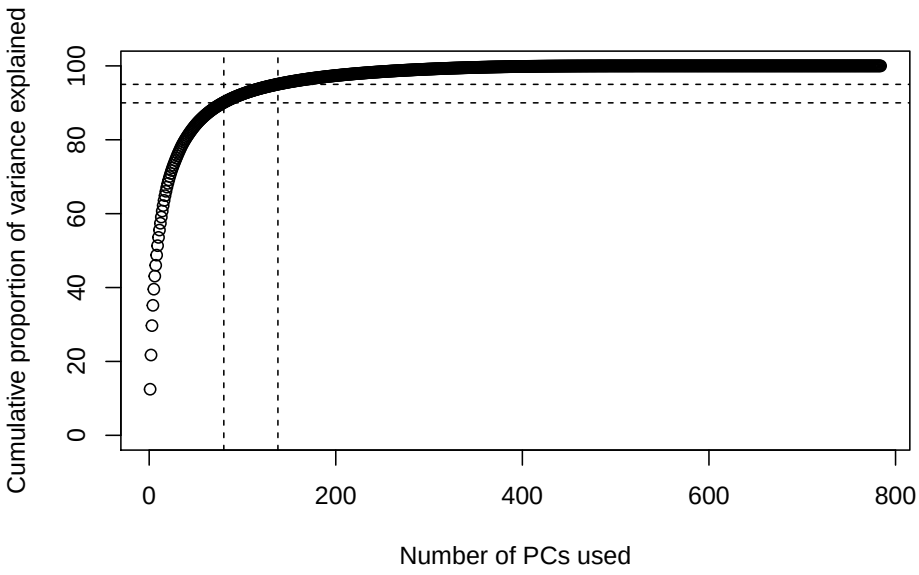
```
plot(mnist3.pca$sdev) # scree plot
```



It can also be useful to plot the cumulative sum of the total proportion of

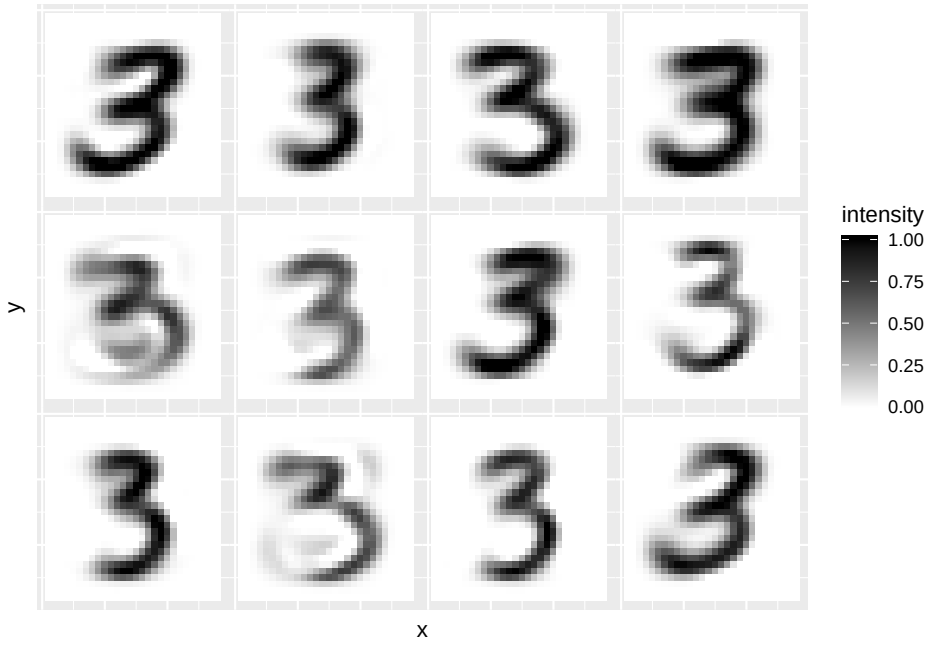
variance explained by a given number of principal components. I've drawn on horizontal lines at 90% and 95% of variance explained, to help identify when we cross these thresholds. We need 80 components to explain 90% of the variance, and 138 components to explain 95% of the variance.

```
cumvar = 100*cumsum(mnist3.pca$sdev^2) / sum(mnist3.pca$sdev^2)
plot(cumvar, ylab="Cumulative proportion of variance explained", xlab="Number of PCs used", ylim=
abline(h=90, lty=2)
abline(v=min(which(cumvar>90)), lty=2)
abline(h=95, lty=2)
abline(v=min(which(cumvar>95)), lty=2)
```

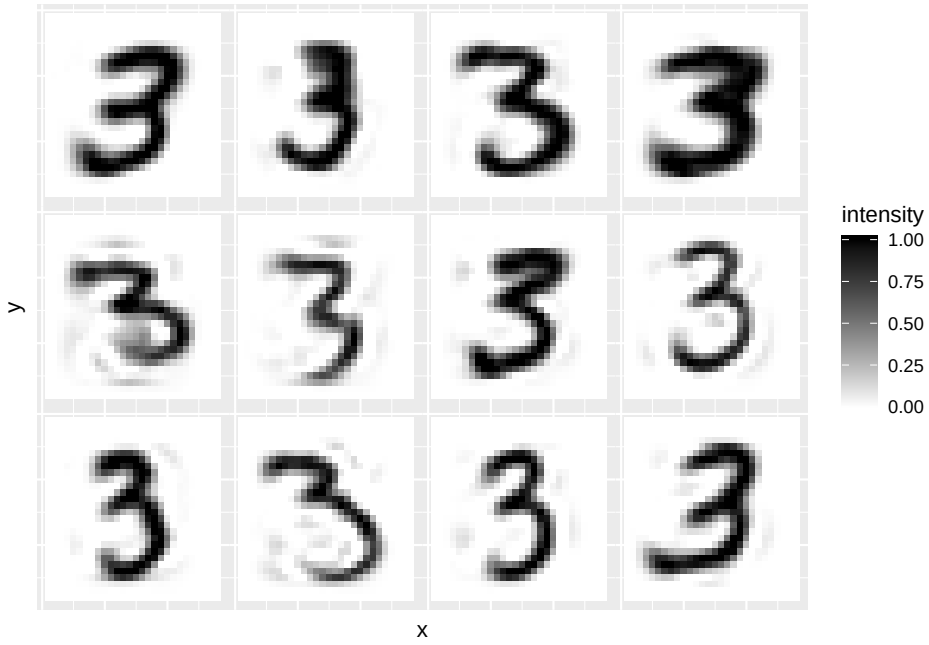


Let's now look at the reconstruction using $r = 10,\ 50,\ 100$ and 500 components to see how the accuracy changes.

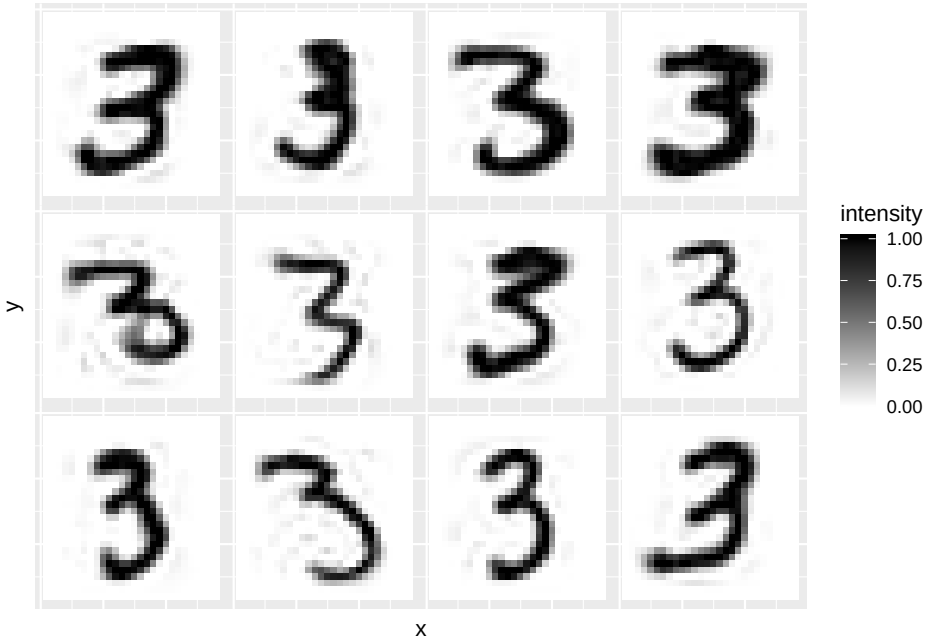
```
r=10
recon =  mnist3.pca$x[,1:r] %*% t(mnist3.pca$rotation[,1:r])
plot.mnist2(matrix(rep(xbar,12), byrow=T, nr=12)+recon[1:12,])
```



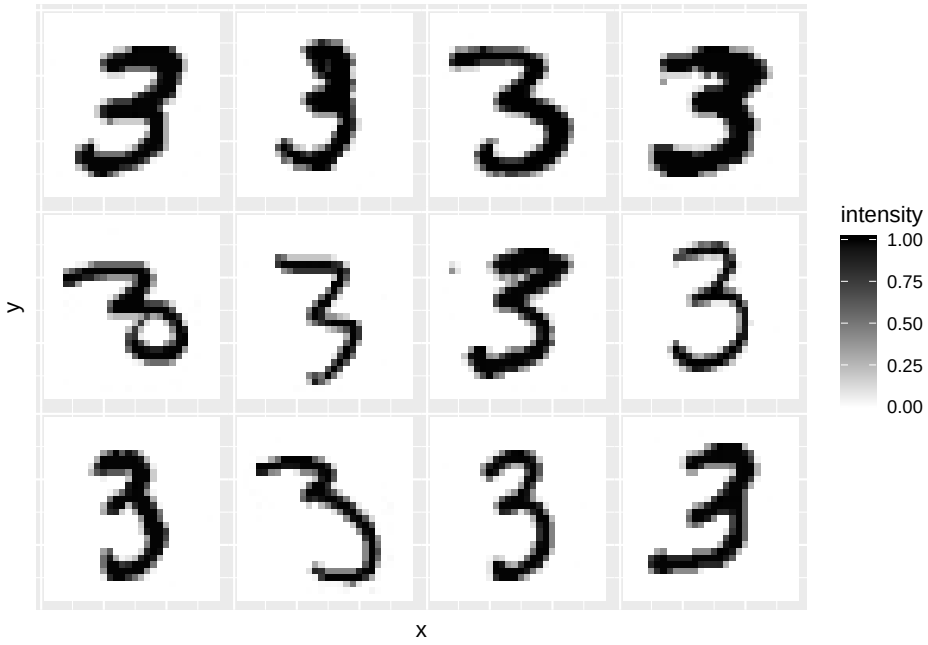

```
r=50
recon =  mnist3.pca$x[,1:r] %*% t(mnist3.pca$rotation[,1:r])
plot.mnist2(matrix(rep(xbar,12), byrow=T, nr=12)+recon[1:12,])
```



```
r=100
recon =  mnist3.pca$x[,1:r] %*% t(mnist3.pca$rotation[,1:r])
plot.mnist2(matrix(rep(xbar,12), byrow=T, nr=12)+recon[1:12,])
```



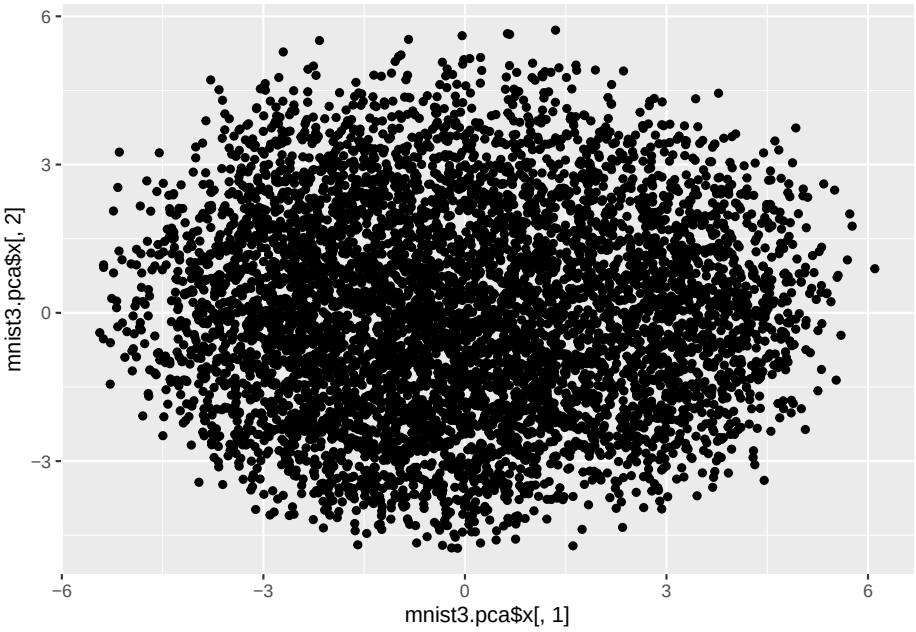

```
r=500
recon =  mnist3.pca$x[,1:r] %*% t(mnist3.pca$rotation[,1:r])
plot.mnist2(matrix(rep(xbar,12), byrow=T, nr=12)+recon[1:12,])
```



We can see that as the number of components increases the reconstructions start to look more like the original 12 images.

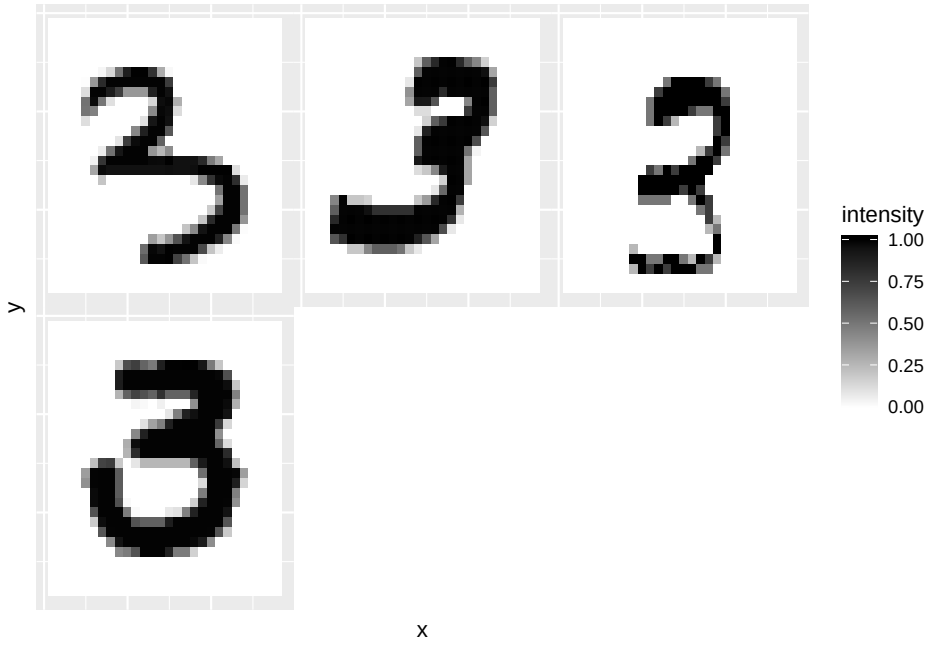
We can visualise the range of 3s by looking at a scatter plot of the first two principal components.

```
library(ggplot2)
qplot(mnist3.pca$x[,1], mnist3.pca$x[,2])
```



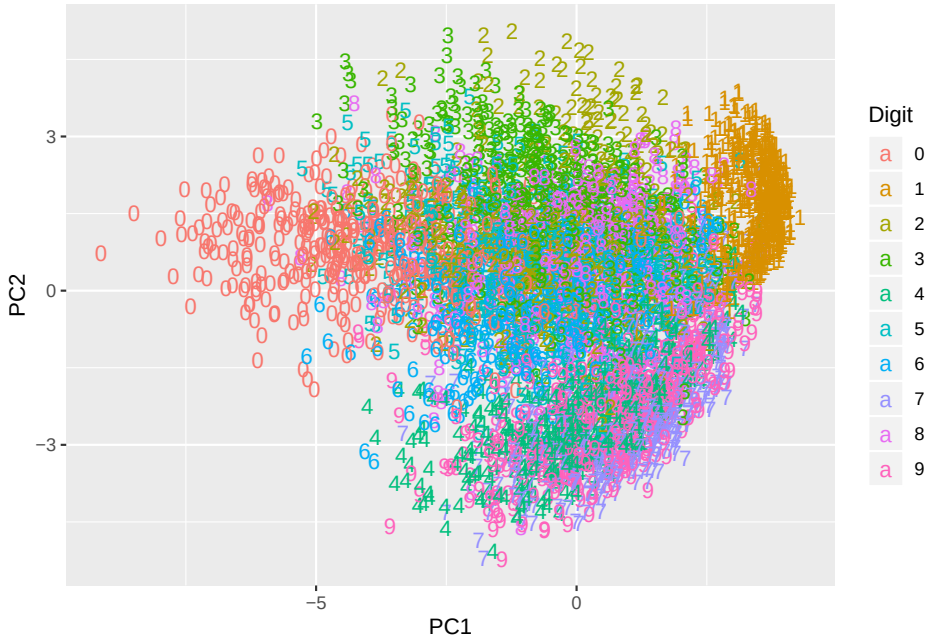
We can then finding images that differ according to these two PC scores. The first plot below is the 3 with the smallest PC1 score, and the second has the largest PC1 score. The third plot has the smallest PC2 score, and the fourth plot the largest PC2 score. These four different 3s differ in more than just the first two principal components, but you can see the effect of the PC1 score is to slant the image forward or backward, whereas PC2 changes the thickness of the line.

```
image_list <- c(which.min(mnist3.pca$x[,1]), which.max(mnist3.pca$x[,1]),
                which.min(mnist3.pca$x[,2]), which.max(mnist3.pca$x[,2]))
plot.mnist(mnist3[image_list,]) # plot the first 12 images
```



Finally, let's do PCA on a selection of the 60,000 images (not just the 3s). You can compute the SVD (which is what `prcomp` uses to do PCA) on a $60,000 \times 784$ matrix, but it takes a long time on most computers, so here I've just computed the first two components on a random selection of 5,000 images using the option `rank=2` which significantly speeds up the computation time.

```
# Note this is slow to compute!
image_index <- sample(1:60000, size=5000) # select a random sample of images
mnist.pca <- prcomp(mnist$train$x[image_index,], rank=2)
Digit = as.factor(mnist$train$y[image_index])
ggplot(as.data.frame(mnist.pca$x), aes(x=PC1, y=PC2, colour=Digit, label=Digit))+
geom_text(aes(label=Digit))
```



We can see from this scatter plot that the first two principal components do a surprisingly good job of separating and clustering the digits.

4.4 Computer tasks

4.4.0.0.1 Exercise 1

Using the `iris` dataset, familiarize yourself with the `prcomp` command and its output.

```
Now, instead of using `prcomp` we will do the analysis ourselves using the `eigen` command.
```

- Start by computing the sample mean and sample variance of the dataset (use $n-1$ as the denominator when you compute the sample variance to get the same answer as provided by `prcomp`).
- Now compute the eigenvalues and eigenvectors of the covariance matrix using `eigen`. Check that these agree with those computed by `prcomp` (noting that `prcomp` returns the standard deviation which is the square root of the eigenvalues).
- Now compute the principal component scores by multiplying $\mathbf{X}$ by the matrix of eigenvectors $\mathbf{V}$. Check your answer agrees with the scores provided by `prcomp`.

Now we will do the same thing again, but using the `svd` command.

- Compute the column centred data matrix $\frac{1}{\sqrt{n-1}}\mathbf{HX}$.

- Compute the SVD of $\frac{1}{\sqrt{n-1}}\mathbf{HX}$ and $\mathbf{HX}$. How are the two sets of singular values related, and how do they relate to the eigenvalues computed previously. Are the singular vectors of $\frac{1}{\sqrt{n-1}}\mathbf{HX}$ and $\mathbf{HX}$ the same?
- Compute the SVD scores by doing both $\mathbf{HXV}$ and $\mathbf{U\Sigma}$, where

$$\mathbf{HX} = \mathbf{U\Sigma V}^\top$$

  is the SVD of $\mathbf{HX}$.

4.4.0.0.2 Exercise 2

In this question we will look at the crabs data from the `MASS` R package. We will focus on the 5 continuous variables, all measured in mm:

- FL = frontal lobe size
- RW = rear width
- CL = carapace length
- CW = carapace width
- BD = body depth.

The sample size is 200.

```
library(MASS)
?crabs # read the help page to find out about the dataset
X=crabs[,4:8]     # construct data matrix X with columns FL, RW, CL, CW, BD
```

- Carry out PCA on the data in X using the covariance matrix, including obtaining a scree plot and plotting the PC scores.

Some questions:

- Do you have any suggestions for an interpretation for the 1st PC?
- Are you able to come up with an interpretation for the 2nd PC?
- Do you think an analysis based on the sample covariance matrix $\mathbf{S}$ or the correlation matrix $\mathbf{R}$ is preferable with this dataset? Note that you can use `scale=TRUE` in `prcomp` to carry out PCA on $\mathbf{R}$. Does it make much difference which is used?
- Without doing any computation, think about what you expect the sample mean and sample covariance matrix to be for the PC scores. Check this numerically.
- Now check other properties of the PC scores listed in proposition 4.2.
- Try the following transformations of the data.
  - adding a constant to the data

$$\mathbf{z} = \mathbf{x} + \mathbf{c},$$

– scaling the data:

$$\mathbf{z} = \mathbf{D}\mathbf{x}$$

for some diagonal matrix $\mathbf{D}$

– rotating the data:

$$\mathbf{z} = \mathbf{U}\mathbf{x}$$

for some $p \times p$ orthogonal matrix $\mathbf{U}$. You can generate a random orthogonal matrix using the following commands

```
library(pracma)
U <- randortho(5)
```

Check the effect of each transformation on the principal components (the loadings/eigenvectors), the principal component scores, and the variance of the principal components (the eigenvalues).

4.4.0.0.3 Exercise 3

Download the final Premier League table for the 2018-19 season from https://www.rotowire.com/soccer/league-table.php?season=2018. There is a button to download the csv (comma separated variable) file in the bottom right hand corner.

- Load the data into R using the command `read.csv`. Note that you may need to manually delete the first row of the csv file before doing this. (*I haven't given you the command here, as learning how to do this yourself is important*).
- Repeat the analysis from section 4.2.2. Does the meaning of the principal components change? Was the 2018-19 league season notably different to the 2019-20 season (which is the season analysed in the notes)?

4.5 Exercises

1. Consider the following data in $\mathbb{R}^2$

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \ \mathbf{x}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \ \mathbf{x}_3 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

- What is the orthogonal projection of these points onto

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and onto

$$\mathbf{u}_2 = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ 2 \end{pmatrix}?$$

- Compute the sample variance matrix of the data points, and compute its spectral decomposition.
- Which unit vector $\mathbf{u}$ would maximize the variance of these projections?
- What vector $\mathbf{u}$ would minimize

$$\sum_{i=1}^{4} ||\mathbf{x}_i - \mathbf{u}\mathbf{u}^\top \mathbf{x}_i||_2^2?$$

  This is the sum of squared errors from a rank 1 approximation to the data.
- Plot the data points and convince yourself that your answers make intuitive sense.

2. Consider a population covariance matrix $\mathbf{\Sigma}$ of the form

$$\mathbf{\Sigma} = \gamma \mathbf{I}_p + \mathbf{a}\mathbf{a}^\top$$

   where $\gamma > 0$ is a scalar, $\mathbf{I}_p$ is the $p \times p$ identity matrix and $\mathbf{a}$ is a vector of dimension p.
   - Show that $\mathbf{a}$ is an eigenvector of $\mathbf{\Sigma}$.
   - Show that if $\mathbf{b}$ is any vector such that $\mathbf{a}^\top \mathbf{b} = 0$, then $\mathbf{b}$ is also an eigenvector of $\mathbf{\Sigma}$.
   - Obtain all the eigenvalues of $\mathbf{\Sigma}$.
   - Determine expressions for the proportion of variability 'explained' by:
   i. the largest (population) principal component of $\mathbf{\Sigma}$;
   ii. the r largest (population) principal components of $\mathbf{\Sigma}$, where $1 < r \leq p$.
3. A covariance matrix has the following eigenvalues:

```
##  [1] 4.22 2.38 1.88 1.11 0.91 0.82 0.58 0.44 0.35 0.19 0.05 0.04 0.04
```

   - Sketch a scree plot.
   - Determine the minimum number of principal components needed to explain 90% of the total variation.
   - Determine the number of principal components whose eigenvalues are above average.

4. Measurements are taken on $p = 3$ variables x_1, x_2 and x_3, with sample correlation matrix

$$\mathbf{R} = \begin{pmatrix} 1 & 0.5792 & 0.2414 \\ 0.5792 & 1 & 0.5816 \\ 0.2414 & 0.5816 & 1 \end{pmatrix}.$$

   The variable z_j is the standardised versions of x_j, $j = 1, 2, 3$, i.e. each z_j has sample mean 0 and variance 1. One observation has $z_1 = z_2 = z_3 = 0$ and a second observation has $z_1 = z_2 = z_3 = 1$. Calculate the three principal component scores for each of these observations.

5. Do exam question 1 part (a) from the 2017-18 exam paper. You will find the past exam papers on Moodle.

Chapter 5

Canonical Correlation Analysis (CCA)

The videos for this chapter are available at the following links

- 5.1: Introduction to CCA
- 5.1.1: The first pair of CC variables
- 5.1.2: Example: Premier league data
- 5.2: The full set of CC variables
- 5.3: Properties of CCA

Suppose we observe a random sample of n bivariate observations

$$\mathbf{z}_1 = (x_1, y_1)^\top, \ldots, \mathbf{z}_n = (x_n, y_n)^\top.$$

If we are interested in exploring possible dependence between the x_i's and y_i's then among the first things we would do would be to obtain a scatterplot of the x_i's against the y_i's and calculate the correlation coefficient. Recall that the sample correlation coefficient is defined by

$$r = \mathbb{C}\,\mathrm{or}(x, y) = \frac{S_{xy}}{\sqrt{S_{xx}}\sqrt{S_{yy}}} \tag{5.1}$$

$$= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\left(\sum_{i=1}^n (x_i - \bar{x})^2\right)^{1/2} \left(\sum_{i=1}^n (y_i - \bar{y})^2\right)^{1/2}} \tag{5.2}$$

where $\bar{x} = n^{-1} \sum_{i=1}^n x_i$ and $\bar{y} = n^{-1} \sum_{i=1}^n y_i$ are the sample means.

Recall that the sample correlation is a **scale-free measure** of the strength of the **linear dependence** between the x_i's and the y_i's.

In this chapter we investigate the multivariate analogue of this question. Instead of our bivariate observations being a pair of scalars, suppose instead that we are

given two different random vectors $\mathbf{x}$ and $\mathbf{y}$. In otherwords, for each subject/case i we have observations $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$.

Multivariate data structures can be understood better if we look at low-dimensional projections of the data. The question is, given a sample $\{\mathbf{x}_i,\ \mathbf{y}_i\}_{i=1}^n$, what is a sensible way to assess and describe the strength of the linear dependence between the two vectors?

Canonical correlation analysis (CCA) gives an answer to this question in terms of the best low-dimensional linear projections of the $\mathbf{x}$ and $\mathbf{y}$ random variables. In a comparable way to PCA, 'best' in CCA is defined in terms of maximizing correlations. A key role is played by the singular value decomposition (SVD) introduced in Chapter 3.

5.1 The first pair of canonical variables

Some notation

Assume we are given a random sample of vectors $\mathbf{x}_i, \mathbf{y}_i$, and that we stack these into a vector $\mathbf{z}_i$

$$\mathbf{z}_i = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{y}_i \end{pmatrix} = (\mathbf{x}_i^\top, \mathbf{y}_i^\top)^\top, \quad i = 1, \ldots, n,$$

where the $\mathbf{x}_i$ are $p \times 1$, the $\mathbf{y}_i$ are $q \times 1$ and, consequently, the $\mathbf{z}_i$ are $(p+q) \times 1$. We are interested in determining the strength of linear association between the $\mathbf{x}_i$ vectors and the $\mathbf{y}_i$ vectors. We formulate this task as an optimisation problem (cf. PCA).

First, some notation:

- Let $\bar{\mathbf{x}}$, $\bar{\mathbf{y}}$, and $\bar{\mathbf{z}}$ denote the sample mean vectors of the $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ respectively.
- Let $\mathbf{S_{zz}}$ denote the sample covariance matrix of the $\mathbf{z}_i$, $i = 1, \ldots, n$. Then $\mathbf{S_{zz}}$ can be written in block matrix form

  $$\mathbf{S_{zz}} = \begin{bmatrix} \mathbf{S_{xx}} & \mathbf{S_{xy}} \\ \mathbf{S_{yx}} & \mathbf{S_{yy}} \end{bmatrix},$$

  where $\mathbf{S_{xx}}$ $(p \times p)$ is the sample covariance matrix of the $\mathbf{x}_i$, $\mathbf{S_{yy}}$ $(q \times q)$ is the sample covariance of the $\mathbf{y}_i$, and the cross-covariance matrices are given by

  $$\overset{p\times q}{\mathbf{S}}_{\mathbf{xy}} = n^{-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{y}_i - \bar{\mathbf{y}})^\top \qquad \text{and} \qquad \overset{q\times p}{\mathbf{S}}_{\mathbf{yx}} = \mathbf{S}_{\mathbf{xy}}^\top.$$

Defining the optimization objective

We want to find the linear combination of the $\mathbf{x}$-variables and the linear combination of the $\mathbf{y}$-variables which is most highly correlated.

One version of the optimisation problem we want to solve is: find non-zero vectors $\overset{p\times 1}{\mathbf{a}}$ and $\overset{q\times 1}{\mathbf{b}}$ which maximise the correlation coefficient

$$\mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y}) = \frac{\mathbf{a}^\top\mathbf{S_{xy}}\mathbf{b}}{(\mathbf{a}^\top\mathbf{S_{xx}}\mathbf{a})^{1/2}(\mathbf{b}^\top\mathbf{S_{yy}}\mathbf{b})^{1/2}}.$$

In other words:

$$\begin{array}{ll} \text{Maximise} & \mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y}), \\ \text{for non-zero vectors} & \mathbf{a}\ \ (p\times 1) \text{ and } \mathbf{b}\ \ (q\times 1) \end{array} \tag{5.3}$$

where $\mathbb{C}\,\mathrm{or}(\cdot,\cdot)$ is defined in (5.2).

Intuitively, this objective makes sense, because we want to find the linear combination of the $\mathbf{x}$-variables and the linear combination of the $\mathbf{y}$-variables which are most highly correlated. However, note that for any $\gamma > 0$ and $\delta > 0$,

$$\mathbb{C}\,\mathrm{or}(\gamma\mathbf{a}^\top\mathbf{x}, \delta\mathbf{b}^\top\mathbf{y}) = \frac{\gamma\delta}{\sqrt{\gamma^2\delta^2}}\mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y}) \tag{5.4}$$

$$= \mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y}). \tag{5.5}$$

$\mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y})$ is invariant to (i.e. unchanged by) multiplication of $\mathbf{a}$ and $\mathbf{b}$ by positive scalars. Consequently there will be an infinite number of solutions to this optimisation problem, because if $\mathbf{a}$ and $\mathbf{b}$ are solutions, then so are $\gamma\mathbf{a}$ and $\delta\mathbf{b}$, for any $\gamma > 0$ and $\delta > 0$.

A more useful way to formulate this optimisation problem is

$$\begin{array}{ll} \text{Maximize} & \mathbf{a}^\top\mathbf{S_{xy}}\mathbf{b} \\ \text{subject to} & \mathbf{a}^\top\mathbf{S_{xx}}\mathbf{a} = \mathbf{b}^\top\mathbf{S_{yy}}\mathbf{b} = 1. \end{array} \tag{5.6}$$

Thankfully, there is a link between the solutions of the two optimization problems (5.3) and (5.6). Firstly, the invariance of $\mathbb{C}\,\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y})$ means that if $\breve{\mathbf{a}}$ and $\breve{\mathbf{b}}$ are a solution to problem (5.6), then for any $\gamma,\ \delta > 0$, we have that $\mathbf{a} = \gamma\breve{\mathbf{a}}$ and $\mathbf{b} = \delta\breve{\mathbf{b}}$ are a solution to (5.3).

Conversely, we can convert any solution to optimization problem (5.3) to be a solution to the problem (5.6):

Proposition 5.1. *If* $\mathbf{a}$ *and* $\mathbf{b}$ *maximise* (5.3)*, then*

$$\breve{\mathbf{a}} = \frac{\mathbf{a}}{(\mathbf{a}^\top\mathbf{S_{xx}}\mathbf{a})^{1/2}} \qquad \textit{and} \qquad \breve{\mathbf{b}} = \frac{\mathbf{b}}{(\mathbf{b}^\top\mathbf{S_{yy}}\mathbf{b})^{1/2}}$$

are a solution to the constrained maximization problem (5.6).

Proof. Suppose $\mathbf{a}$ and $\mathbf{b}$ are solutions to optimization problem (5.3). Then invariance with respect to rescaling implies that $\check{\mathbf{a}} = \mathbf{a}/(\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a})^{1/2}$ and $\check{\mathbf{b}} = \mathbf{b}/(\mathbf{b}^\top \mathbf{S_{yy}} \mathbf{b})^{1/2}$ also achieve the optima. But $\check{\mathbf{a}}$ and $\check{\mathbf{b}}$ satisfy the constraints $\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a} = \mathbf{b}^\top \mathbf{S_{yy}} \mathbf{b} = 1$ because

$$\check{\mathbf{a}}^\top \mathbf{S_{xx}} \check{\mathbf{a}} = \frac{\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a}}{\left\{(\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a})^{1/2}\right\}^2} = \frac{\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a}}{\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a}} = 1$$

and similarly for $\check{\mathbf{b}}$. So $\check{\mathbf{a}}$ and $\check{\mathbf{b}}$ maximises (5.6) subject to the constraints. □

5.1.1 The first canonical components

As in the chapter on PCA, the optimal solution for CCA can be computed using the singular value decomposition. Before we describe the result, let's prove the following proposition from Chapter 3

Proposition 5.2. *For any matrix* $\mathbf{Q}$*, we have*

$$\max_{\mathbf{a},\mathbf{b}:\, ||\mathbf{a}||=||\mathbf{b}||=1} \mathbf{a}^\top \mathbf{Q} \mathbf{b} = \sigma_1.$$

with the maximum obtained at $\mathbf{a} = \mathbf{u}_1$ *and* $\mathbf{b} = \mathbf{v}_1$*, the first left and right singular vectors of* $\mathbf{Q}$*.*

Proof. We'll use the method of Lagrange multipliers to prove this result. Consider the objective

$$\mathcal{L} = \mathbf{a}^\top \mathbf{Q} \mathbf{b} + \frac{\lambda_1}{2}(1 - \mathbf{a}^\top \mathbf{a}) + \frac{\lambda_2}{2}(1 - \mathbf{b}^\top \mathbf{b}).$$

The factor of $1/2$ is there to simplify the maths once we differentiate. Differentiating with respect to $\mathbf{a}$ and $\mathbf{b}$ and setting the derivative equal to zero gives

$$\mathbf{0} = \mathbf{Q}\mathbf{b} - \lambda_1 \mathbf{a} \tag{5.7}$$

$$\mathbf{0} = \mathbf{Q}^\top \mathbf{a} - \lambda_2 \mathbf{b} \tag{5.8}$$

where for the second equation we've noted that $\mathbf{a}^\top \mathbf{Q} \mathbf{b} = \mathbf{b}^\top \mathbf{Q}^\top \mathbf{a}$. Left multiplying the first equation by $\mathbf{a}^\top$ and the second by $\mathbf{b}^\top$, and recalling that $\mathbf{a}^\top \mathbf{a} = \mathbf{b}^\top \mathbf{b} = 1$, shows that the two Lagrange multipliers are the same $\lambda_1 = \lambda_2 =: \lambda$ say.

Substituting $\mathbf{a} = \mathbf{Q}\mathbf{b}/\lambda$ into (5.8) gives

$$\mathbf{Q}^\top \mathbf{Q} \mathbf{b} = \lambda^2 \mathbf{b},$$

and so we can see that $\mathbf{b}$ is an eigenvector of $\mathbf{Q}^\top \mathbf{Q}$, and thus we must have $\mathbf{b} = \mathbf{v}_i$ for some i, i.e., $\mathbf{b}$ is one of the right singular vectors of $\mathbf{Q}$. Similarly, substituting $\mathbf{b} = \mathbf{Q}^\top \mathbf{a}/\lambda$ into (5.7) gives

$$\mathbf{Q}\mathbf{Q}^\top \mathbf{a} = \lambda^2 \mathbf{a}.$$

So $\mathbf{a} = \mathbf{u}_j$ for some j, i.e., $\mathbf{a}$ is one of the left singular vectors of $\mathbf{Q}$.

Finally, consider the original objective with $\mathbf{a} = \mathbf{u}_j$ and $\mathbf{b} = \mathbf{v}_i$:

$$\mathbf{u}_j^\top \mathbf{Q}\mathbf{v}_i = \sigma_i \mathbf{u}_j^\top \mathbf{u}_i = \begin{cases} \sigma_i & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Hence we minimize the objective by taking $\mathbf{a} = \mathbf{u}_1$ and $\mathbf{b} = \mathbf{v}_1$, and then we find

$$\max_{\mathbf{a},\mathbf{b}:\, ||\mathbf{a}||=||\mathbf{b}||=1} \mathbf{a}^\top \mathbf{Q}\mathbf{b} = \sigma_1.$$

□

Main result

We're now in a position to describe the main result giving the first pair of canonical variables

Proposition 5.3. *Assume that* $\mathbf{S_{xx}}$ *and* $\mathbf{S_{yy}}$ *are both non-singular, and consider the singular value decomposition of the matrix* $\mathbf{Q} := \mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}\mathbf{S_{yy}}^{-1/2}$

$$\mathbf{Q} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{j=1}^{t} \sigma_j \mathbf{u}_j \mathbf{v}_j^\top, \tag{5.9}$$

where $t = \text{rank}(\mathbf{Q})$ *and* $\sigma_1 \geq \cdots \geq \sigma_t > 0$.

Then the solution to the constrained optimization problem (5.6) *is*

$$\mathbf{a} = \mathbf{S_{xx}}^{-1/2}\mathbf{u}_1 \quad \textit{and} \quad \mathbf{b} = \mathbf{S_{yy}}^{-1/2}\mathbf{v}_1.$$

The maximum value of the correlation coefficient is given by the largest singular value σ_1*:*

$$\max_{\mathbf{a},\mathbf{b}} \mathbb{C}\text{or}(\mathbf{a}^\top \mathbf{x}, \mathbf{b}^\top \mathbf{b}) = \sigma_1.$$

Proof. If we let

$$\tilde{\mathbf{a}} = \mathbf{S_{xx}}^{1/2}\mathbf{a} \qquad \text{and} \qquad \tilde{\mathbf{b}} = \mathbf{S_{yy}}^{1/2}\mathbf{b},$$

we may write the constraints $\mathbf{a}^\top \mathbf{S_{xx}}\mathbf{a} = \mathbf{b}^\top \mathbf{S_{yy}}\mathbf{b} = 1$ as

$$\tilde{\mathbf{a}}^\top \tilde{\mathbf{a}} = 1 \qquad \text{and} \qquad \tilde{\mathbf{b}}^\top \tilde{\mathbf{b}} = 1.$$

Because $\mathbf{S_{xx}}$ and $\mathbf{S_{yy}}$ are non-singular, $\mathbf{S_{xx}}^{1/2}$ and $\mathbf{S_{yy}}^{1/2}$ must also be non-singular, and so we can compute $\mathbf{S_{xx}}^{-1/2}$ and $\mathbf{S_{yy}}^{-1/2}$, using the matrix square roots defined in Section 3.2.3.

If we write

$$\mathbf{a} = \mathbf{S_{xx}}^{-1/2}\tilde{\mathbf{a}} \qquad \text{and} \qquad \mathbf{b} = \mathbf{S_{yy}}^{-1/2}\tilde{\mathbf{b}},$$

then the optimisation problem (5.6) becomes

$$\max_{\tilde{\mathbf{a}},\tilde{\mathbf{b}}} \tilde{\mathbf{a}}^\top \mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{S}_{\mathbf{xy}}\mathbf{S}_{\mathbf{yy}}^{-1/2}\tilde{\mathbf{b}}$$

subject to

$$||\tilde{\mathbf{a}}|| = 1 \qquad \text{and} \qquad ||\tilde{\mathbf{b}}|| = 1.$$

Then by Proposition 5.2 we can see that

$$\tilde{\mathbf{a}} = \mathbf{u}_1 \quad \text{and} \quad \tilde{\mathbf{b}} = \mathbf{v}_1$$

and the result follows.

□

We will label the solution found as

$$\mathbf{a}_1 := \mathbf{S}_{xx}^{-\frac{1}{2}}\mathbf{u}_1 \qquad \text{and} \qquad \mathbf{b}_1 := \mathbf{S}_{yy}^{-\frac{1}{2}}\mathbf{v}_1$$

to stress that $\mathbf{a}_1$ and $\mathbf{b}_1$ are the **first** pair of **canonical correlation (CC) vectors**. The variables $\eta_1 = \mathbf{a}_1^\top(\mathbf{x} - \bar{\mathbf{x}})$ and $\psi_1 = \mathbf{b}_1^\top(\mathbf{y} - \bar{\mathbf{y}})$ are called the first pair of **canonical correlation variables**, and $\sigma_1 = \mathbb{C}\text{or}(\eta_1, \psi_1)$ is the **first canonical correlation**.

5.1.2 Example: Premier league football

Lets again return to the Premier League from the previous chapter.

```
library(dplyr)
prem1920 <- read.csv('data/2019_2020EPL.csv')
# the data can be downloaded from https://www.rotowire.com/soccer/league-table.php
table <- prem1920 %>% dplyr::select(Team, W, D, L, G, GA, GD)
knitr::kable(head(table,5), booktabs = TRUE, escape=FALSE)
```

Team	W	D	L	G	GA	GD
Liverpool	32	3	3	85	33	52
Manchester City	26	3	9	102	35	67
Manchester United	18	12	8	66	36	30
Chelsea	20	6	12	69	54	15
Leicester City	18	8	12	67	41	26

We shall treat W and D, the number of wins and draws, as the $\mathbf{x}$-variables. The number of goals for and against, G and GA, will be treated as the $\mathbf{y}$-variables. The number of losses and the goal difference, L and GD, are omitted as they provide no additional information when we know W and D.

We shall consider the questions

- how strongly associated are the match outcome variables, W and D, with the goals for and against variables, G and GA?

- what linear combination of W and D, and of G and GA are most strongly correlated?

Firstly, we need to compute the three covariance matrices needed for CCA. These are easily computed in R:

```
X <- table[,c('W','D')] # W and D
Y <- table[,c('G','GA')] # G and GA
S_xx <- cov(X)
S_yy <- cov(Y)
S_xy <- cov(X,Y)
```

giving

$$\mathbf{S}_{xx} = \begin{pmatrix} 40.4 & -9.66 \\ -9.66 & 10.7 \end{pmatrix}, \quad \mathbf{S}_{yy} = \begin{pmatrix} 354 & -155 \\ -155 & 141 \end{pmatrix}, \quad \mathbf{S}_{xy} = \mathbf{S}_{yx}^\top = \begin{pmatrix} 108 & -60 \\ -28.9 & -2.36 \end{pmatrix}.$$

We now want to calculate the matrix $\mathbf{Q}$ in (5.9) and then find its singular valued decomposition. We first need to find $\mathbf{S_{xx}}^{-1/2}$ and $\mathbf{S_{yy}}^{-1/2}$. Using R to do the computations we obtain the spectral decompositions

```
eigen_xx <- eigen(S_xx)
Vx <- eigen_xx$vectors
S_xx_invsqrt <- Vx %*% diag(1/sqrt(eigen_xx$values)) %*% t(Vx)
# check S_xx %*% S_xx_invsqrt %*% S_xx_invsqrt is the identity matrix
```

$$\mathbf{S}_{xx} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top = \begin{pmatrix} -0.959 & -0.285 \\ 0.285 & -0.959 \end{pmatrix} \begin{pmatrix} 43.2 & 0 \\ 0 & 7.82 \end{pmatrix} \begin{pmatrix} -0.959 & 0.285 \\ -0.285 & -0.959 \end{pmatrix}$$

and so

$$\begin{aligned} \mathbf{S_{xx}}^{-1/2} &= \mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^\top \\ &= \begin{pmatrix} -0.959 & -0.285 \\ 0.285 & -0.959 \end{pmatrix} \begin{pmatrix} 0.152 & 0 \\ 0 & 0.357 \end{pmatrix} \begin{pmatrix} -0.959 & 0.285 \\ -0.285 & -0.959 \end{pmatrix} \\ &= \begin{pmatrix} 0.169 & 0.0561 \\ 0.0561 & 0.341 \end{pmatrix}. \end{aligned}$$

Similarly, we find

```
eigen_yy <- eigen(S_yy)
Vy <- eigen_yy$vectors
S_yy_invsqrt <- Vy %*% diag(1/sqrt(eigen_yy$values)) %*% t(Vy)
```

$$\mathbf{S}_{\mathbf{yy}}^{-1/2} = \begin{pmatrix} 0.0657 & 0.0337 \\ 0.0337 & 0.112 \end{pmatrix}.$$

Consequently,

$$\begin{aligned} \mathbf{Q} &= \mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{S}_{\mathbf{xy}}\mathbf{S}_{\mathbf{yy}}^{-1/2} \\ &= \begin{pmatrix} 0.169 & 0.0561 \\ 0.0561 & 0.341 \end{pmatrix} \begin{pmatrix} 108 & -60 \\ -28.9 & -2.36 \end{pmatrix} \begin{pmatrix} 0.169 & 0.0561 \\ 0.0561 & 0.341 \end{pmatrix} \\ &= \begin{pmatrix} 0.747 & -0.588 \\ -0.39 & -0.595 \end{pmatrix}. \end{aligned}$$

The SVD of $\mathbf{Q}$ is given by

$$\begin{aligned} \mathbf{Q} &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \\ &= \begin{pmatrix} -0.99 & -0.143 \\ -0.143 & 0.99 \end{pmatrix} \begin{pmatrix} 0.955 & 0 \\ 0 & 0.705 \end{pmatrix} \begin{pmatrix} -0.715 & -0.699 \\ 0.699 & -0.715 \end{pmatrix}^\top \end{aligned} \tag{5.10}$$

So the 1st CC coefficient is 0.955, which is close to its maximum value of 1. The 1st CC weight vectors are given by

```
a1 = S_xx_invsqrt%*% Q_svd$u[,1]
b1 = S_yy_invsqrt%*% Q_svd$v[,1]
```

$$\begin{aligned} \mathbf{a}_1 &= \mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{u}_1 \\ &= \begin{pmatrix} 0.169 & 0.0561 \\ 0.0561 & 0.341 \end{pmatrix} \begin{pmatrix} -0.99 \\ -0.143 \end{pmatrix} \\ &= \begin{pmatrix} -0.175 \\ -0.104 \end{pmatrix} \\ \mathbf{b}_1 &= \mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{v}_1 \\ &= \begin{pmatrix} -0.0234 \\ 0.0541 \end{pmatrix} \end{aligned}$$

This leads to the first pair of CC variables, obtained using these CC vectors/weights:

$$\eta_1 = -0.175(W - \bar{W}) + -0.104(D - \bar{D})$$

and

$$\psi_1 = -0.0234(G - \bar{G}) + 0.0541(GA - \bar{GA}).$$

We can see that ψ_1 is measuring something similar to goal difference $G - GA$, as usually defined, but it gives higher weight to goals conceded than goals scored (0.0541 versus 0.0234).

η_1 is measuring something similar to number of points $3W + D$, as usually defined, but the ratio of points for a win to points for a draw is lower, at around 2:1, as opposed to the usual ratio 3:1.

The full list of the first canonical correlation variables is thus

```
Xcent <- sweep(X,2, colMeans(X)) # column centre the matrix
Ycent <- sweep(Y,2, colMeans(Y)) # column centre the matrix
eta = as.matrix(Xcent)%*%a1
psi = as.matrix(Ycent)%*%b1
```

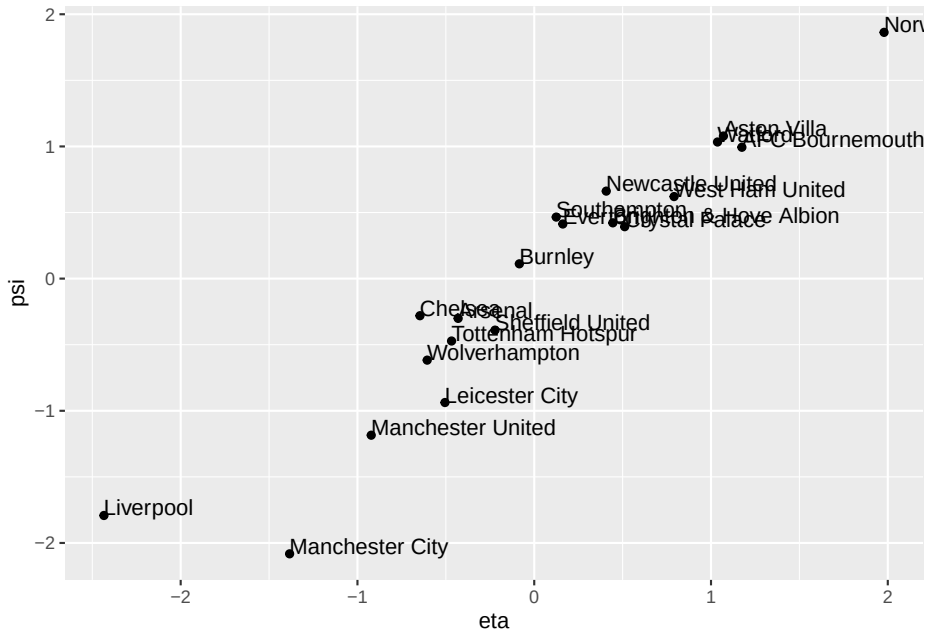
Team	eta	psi
Liverpool	-2.4340723	-1.7921117
Manchester City	-1.3838590	-2.0820965
Manchester United	-0.9221199	-1.1846733
Chelsea	-0.6464941	-0.2807761
Leicester City	-0.5049886	-0.9374949

A scatter plot of the two canonical correlation variables shows the strong correlation between them.

```
cca.out <- data.frame(Team=table$Team, eta=eta, psi=psi)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
ggplot(cca.out, aes(x= eta, y= psi, label=Team))+   geom_point() +
geom_text(aes(label=Team),hjust=0, vjust=0, size=4)
```



We can also do this with the `CCA` package in R. The command `matcor(X,Y)` gives the correlation matrices, and the command `cc` performs CCA. See if you can find the outputs computed above in the output of the `cc` command.

```
library(CCA)
prem.cca <- cc(X,Y)
prem.cca$cor # the canonical correlations
```

```
## [1] 0.9550749 0.7054825
```

```
prem.cca$xcoef # the canonical correlation vectors for X
```

```
##         [,1]      [,2]
## W -0.1750356 0.0313256
## D -0.1042828 0.3293057
```

```
prem.cca$ycoef  # the canonical correlation vectors for Y
```

```
##           [,1]        [,2]
## G  -0.02342507 -0.07003113
## GA  0.05412069 -0.10372100
```

```
head(prem.cca$scores$xscores) # the canonical correlation variables
```

```
##            [,1]       [,2]
## [1,] -2.4340723 -1.4903651
## [2,] -1.3838590 -1.6783187
## [3,] -0.9221199  1.0348282
```

```
## [4,] -0.6464941 -0.8783550
## [5,] -0.5049886 -0.2823947
## [6,] -0.4677660  0.6428713
```

```
head(prem.cca$scores$yscores) # the canonical correlation variables
```

```
##             [,1]        [,2]
## [1,] -1.7921117 -0.39245373
## [2,] -2.0820965 -1.79042487
## [3,] -1.1846733  0.62697465
## [4,] -0.2807761 -1.45009678
## [5,] -0.9374949  0.03833851
## [6,] -0.4722204 -0.16380075
```

```
#prem.cca
#plt.cc(prem.cca, var.label = TRUE, ind.names = table$Team)
```

5.2 The full set of canonical correlations

Let us first recap what we did in the previous section: we found a linear combination of the $\mathbf{x}$-variables and a linear combination of the $\mathbf{y}$-variables which maximised the correlation, and expressed the answer in terms of quantities which arise in the SVD of $\mathbf{Q}$, where

$$\mathbf{Q} \equiv \mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}\mathbf{S_{yy}}^{-1/2} = \mathbf{U\Sigma V}^\top = \sum_{j=1}^{t} \sigma_j \mathbf{u}_j \mathbf{v}_j^\top .$$

We found the maximum value of the correlation $\mathbb{C}\mathrm{or}(\mathbf{a}^\top\mathbf{x}, \mathbf{b}^\top\mathbf{y})$ to be σ_1, achieved using the linear combinations $\eta_1 = \mathbf{a}_1^\top\mathbf{x}$ and $\psi_1 = \mathbf{b}_1^\top\mathbf{y}$ with

$$\mathbf{a}_1 = \mathbf{S_{xx}}^{-1/2}\mathbf{u}_1 \quad \text{and } \mathbf{b}_1 = \mathbf{S_{yy}}^{-1/2}\mathbf{v}_1.$$

We now repeat this process to find the next most important linear combination, subject to being uncorrelated with the first linear combination, as we did with PCA. For $\mathbf{a}^\top\mathbf{x}$ to be uncorrelated with $\eta_1 = \mathbf{a}_1^\top\mathbf{x}$ we require

$$0 = \mathbb{C}\mathrm{ov}(\mathbf{a}_1^\top\mathbf{x}, \mathbf{a}^\top\mathbf{x}) = \mathbf{a}_1^\top\mathbf{S}_{xx}\mathbf{a},$$

and similarly we require the condition $\mathbf{b}_1^\top\mathbf{S}_{yy}\mathbf{b} = 0$ for $\mathbf{b}$.

Thus, to obtain the second canonical correlation coefficient, plus the associated sets of canonical correlation vectors and variables, we need to solve the following optimisation problem:

$$\max_{\mathbf{a},\,\mathbf{b}} \mathbf{a}^\top\mathbf{S_{xy}}\mathbf{b} \tag{5.11}$$

subject to the constraints

$$\mathbf{a}^\top \mathbf{S_{xx}} \mathbf{a} = \mathbf{b}^\top \mathbf{S_{yy}} \mathbf{b} = 1, \tag{5.12}$$

$$\mathbf{a}_1^\top \mathbf{S_{xx}} \mathbf{a} = \mathbf{b}_1^\top \mathbf{S_{yy}} \mathbf{b} = 0. \tag{5.13}$$

Note that maximising (5.11) subject to (5.12) and (5.13)is very similar to the optimisation problem (5.6) considered in the previous section. What is new are the constraints (5.13), which take into account that we have already found the first canonical correlation.

It will probably not surprise you to find that the solution is

$$\mathbf{a}^\top \mathbf{S}_{xy} \mathbf{b} = \sigma_2 \quad \text{achieved at} \quad \mathbf{a} = \mathbf{a}_2 := S_{xx}^{-1/2} \mathbf{u}_2 \text{ and } \mathbf{b} = \mathbf{b}_2 := S_{yy}^{-1/2} \mathbf{v}_2$$

where σ_2 is the second largest singular value of $\mathbf{Q}$, and $\mathbf{u}_2$ and $\mathbf{v}_2$ are the corresponding left and right singular vectors.

Main results

We now state the result in its full generality.

Proposition 5.4. *For $k = 1, \ldots, r = \operatorname{rank}(\mathbf{S}_{xy})$, the solution to sequence of optimization problems*

$$\begin{aligned} \textit{Maximize} \quad & \mathbf{a}^\top \mathbf{S}_{xy} \mathbf{b} && (5.14)\\ \textit{subject to} \quad & \mathbf{a}^\top \mathbf{S}_{xx} \mathbf{a} = \mathbf{b}^\top \mathbf{S}_{yy} \mathbf{b} = 1 && (5.15)\\ \textit{and} \quad & \mathbf{a}_i^\top \mathbf{S}_{xx} \mathbf{a} = \mathbf{b}_i^\top \mathbf{S}_{yy} \mathbf{b} = 0 \textit{ for } i = 1, \ldots, k-1 && (5.16) \end{aligned}$$

is achieved at $\mathbf{a}_k = \mathbf{S}_{xx}^{-1/2} \mathbf{u}_k$ and $\mathbf{b}_k = \mathbf{S}_{yy}^{-1/2} \mathbf{v}_k$ with $\mathbf{a}_k \mathbf{S}_{xy} \mathbf{b}_k = \sigma_k$.

Note that an equivalent way of writing down the problem is as

$$\begin{aligned} \text{Maximize} \quad & \operatorname{tr}(\mathbf{A}^\top \mathbf{S}_{xy} \mathbf{B}) = \sum_{i=1}^{k} \mathbf{a}_i^\top \mathbf{S}_{xy} \mathbf{b}_i \\ \text{subject to} \quad & \mathbf{A}^\top \mathbf{S}_{xx} \mathbf{A} = \mathbf{I} \\ \text{and} \quad & \mathbf{B}^\top \mathbf{S}_{yy} \mathbf{B} = \mathbf{I} \end{aligned}$$

which is in the form of the general problem given in Equation (4.1) if

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \ldots & \mathbf{a}_k \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \mathbf{b}_1 & \ldots & \mathbf{b}_k \end{pmatrix}$$

are matrices containing the canonical correlation vectors as columns.

Before we prove this result, we first give an extension of Proposition 5.2.

Proposition 5.5. *Let* $\mathbf{Q}$ *be an arbitrary matrix with SVD* $\mathbf{Q} = \sum_{k=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$. *For* $k = 1, \ldots, r$ *the solution to the optimization problem*

$$\begin{aligned} &\textit{Maximize} && \mathbf{a}^\top \mathbf{Q} \mathbf{b} && (5.17)\\ &\textit{subject to} && \mathbf{a}^\top \mathbf{a} = \mathbf{b}^\top \mathbf{b} = 1 && (5.18)\\ & && \mathbf{a}_i^\top \mathbf{a} = \mathbf{b}_i^\top \mathbf{b} = 0 \textit{ for } i = 1, \ldots, k-1 && (5.19) \end{aligned}$$

is achieved at

$$\mathbf{a}_k = \mathbf{u}_k, \qquad \mathbf{b}_k = \mathbf{v}_k$$

with

$$\mathbf{a}_k^\top \mathbf{Q} \mathbf{b}_k = \sigma_k.$$

Proof. We will work through the proof of this proposition in the Exercises. □

Proof. **Proof of Proposition 5.4**. We note as before that if we write $\tilde{\mathbf{a}}_j = \mathbf{S}_{\mathbf{xx}}^{1/2} \mathbf{a}_j$ and $\tilde{\mathbf{b}}_j = \mathbf{S}_{\mathbf{yy}}^{1/2} \mathbf{b}_j$, then the constraints become

$$\tilde{\mathbf{a}}^\top \tilde{\mathbf{a}} = \tilde{\mathbf{b}}^\top \tilde{\mathbf{b}} = 1 \text{ and } \tilde{\mathbf{a}}_i^\top \tilde{\mathbf{a}} = \tilde{\mathbf{b}}_i^\top \tilde{\mathbf{b}} = 0 \text{ for } i = 1, \ldots, k.$$

Consequently, we may view constraints (5.13) as corresponding to orthogonality constraints (cf. PCA) in modified coordinate systems.

The objective $\mathbf{a}^\top \mathbf{S}_{xy} \mathbf{b}$ becomes $\tilde{\mathbf{a}}^\top \mathbf{Q} \tilde{\mathbf{b}}$ with

$$\mathbf{Q} = \mathbf{S}_{xx}^{-1/2} \mathbf{S}_{xy} \mathbf{S}_{yy}^{-1/2}.$$

Thus applying Proposition 5.5 gives the desired result. □

To summarize:

- The k^{th} **canonical correlation** is σ_k, the k^{th} largest singular value of $\mathbf{Q}$.
- The k^{th} **canonical correlation vectors** (sometimes called the **weights** for the $\mathbf{x}$ and $\mathbf{y}$ variables) are

$$\mathbf{a}_k = \mathbf{S}_{xx}^{-1/2} \mathbf{u}_k, \qquad \mathbf{b}_k = \mathbf{S}_{yy}^{-1/2} \mathbf{v}_k$$

- The k^{th} **canonical correlation variables** (or **canonical correlation scores**) are

$$\eta_{ik} = \mathbf{a}_k^\top (\mathbf{x}_i - \bar{\mathbf{x}}), \qquad \psi_{ik} = \mathbf{b}_k^\top (\mathbf{y}_i - \bar{\mathbf{y}}).$$

We define the CC variable/score vectors to be

$$\boldsymbol{\eta}_k = (\eta_{1k}, \ldots, \eta_{nk})^\top \text{ and } \boldsymbol{\psi}_k = (\psi_{1k}, \ldots, \psi_{nk})^\top.$$

5.2.1 Example continued

From (5.10), it is seen that the 2nd CC coefficient is given by $\sigma_2 = 0.705$. So the correlation between the second pair of CC variables is smaller than the 1st CC coefficient, though still appreciably different from 0. We now calculate the 2nd CC weight vectors:

$$\mathbf{a}_2 = \mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{q}_2 = \begin{pmatrix} 0.0313 \\ 0.329 \end{pmatrix} \qquad \text{and} \qquad \mathbf{b}_2 = \mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{r}_2 = \begin{pmatrix} -0.07 \\ -0.104 \end{pmatrix},$$

with new variables

$$\eta_2 = 0.0313(W - \bar{W}) + 0.329(D - \bar{D})$$

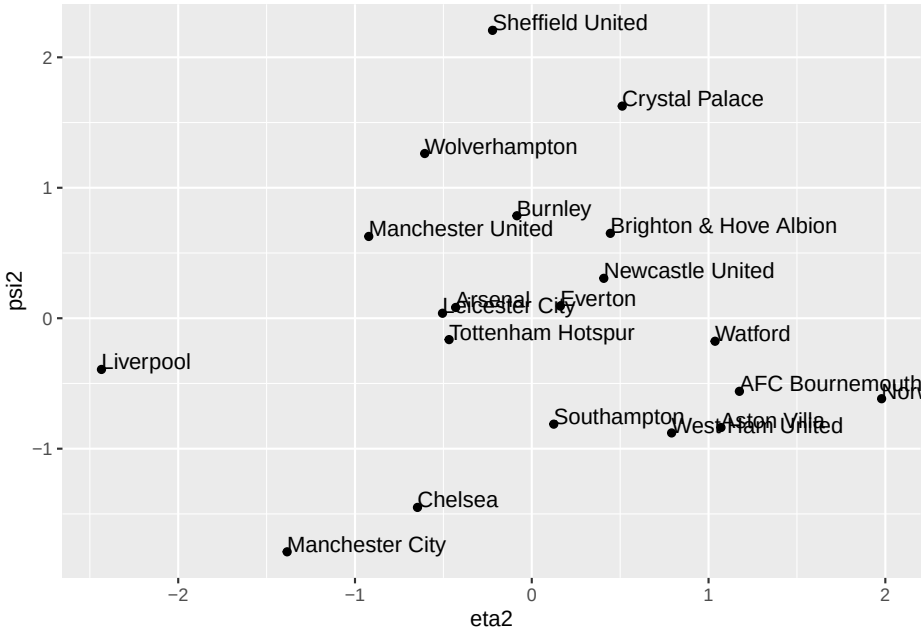
and

$$\psi_2 = -0.07(G - \bar{G}) - 0.104(GA - \bar{GA}).$$

Note that, to a good approximation, η_2 is measuring something similar to the number of draws and, approximately, ψ_2 is something related to the negative of total number of goals in a team's games. So large ψ_2 means relatively few goals in a team's games, and small (i.e. large negative) ψ_2 means a relatively large number of goals in a team's games.

Interpretation of the 2nd CC: teams that have a lot of draws tend to be in low-scoring games and/or teams that have few draws tend to be in high-scoring games.

```
eta2 = prem.cca$scores$xscores[,1]
psi2= prem.cca$scores$yscores[,2]
cca.out <- data.frame(Team=table$Team, eta2=eta2, psi2=psi2)
library(ggplot2)
ggplot(cca.out, aes(x= eta2, y= psi2, label=Team))+   geom_point() +
geom_text(aes(label=Team),hjust=0, vjust=0, size=4)
```



5.3 Properties

The CC variables have a mean of zero. Their correlation structure is given in the following proposition:

Proposition 5.6. *Assume that* $\mathbf{S_{xx}}$ *and* $\mathbf{S_{yy}}$ *both have full rank. Then for* $1 \leq k, \ell \leq t$,

$$\mathbb{C}\,\mathrm{or}(\eta_k, \psi_\ell) = \begin{cases} \sigma_k & \text{if} \quad k = \ell \\ 0 & \text{if} \quad k \neq \ell, \end{cases}$$

and

$$\mathbb{C}\,\mathrm{or}(\eta_j, \eta_k) = \mathbb{C}\,\mathrm{or}(\psi_j, \psi_k) = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

Proof. You will prove this in the exercises. □

5.3.1 Connection with linear regression when $q = 1$

Although CCA is clearly a different technique to linear regression, it turns out that when either $\dim \mathbf{x} = p = 1$ or $\dim \mathbf{y} = q = 1$, there is a close connection between the two approaches.

Consider the case $q = 1$ and $p > 1$. Hence there is only a single y-variable but we still have $p > 1$ x-variables.

Let's make the following assumptions:

1. The $\mathbf{x}_i$ have been centred so that $\bar{\mathbf{x}} = \mathbf{0}_p$, the zero vector.
2. The covariance matrix for the x-variables, $\mathbf{S_{xx}}$, has full rank p.

The first assumption means that

$$\mathbf{S}_{xx} = \frac{1}{n}\mathbf{X}^\top\mathbf{X} \quad \text{and} \quad \mathbf{S}_{xy} = \frac{1}{n}\mathbf{X}^\top\mathbf{y},$$

and the second means that $(\mathbf{X}^\top\mathbf{X})^{-1}$ exists.

Since $q = 1$, the matrix we decompose in CCA

$$\mathbf{Q} = \mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}\mathbf{S}_{yy}^{-1/2}$$

is a $p \times 1$ vector. Consequently, its SVD is just

$$\mathbf{Q} = \sigma_1\mathbf{u}_1,$$

where

$$\sigma_1 = ||\mathbf{Q}||_F = (\mathbf{Q}^\top\mathbf{Q})^{\frac{1}{2}} \qquad \text{and} \qquad \mathbf{u}_1 = \mathbf{Q}/||\mathbf{Q}||_F.$$

Note that $\mathbf{v} = 1$ here. Consequently, the first canoncial correlation vector for $\mathbf{x}$ is

$$\begin{aligned}
\mathbf{a} = \mathbf{S_{xx}}^{-1/2}\mathbf{u}_1 &= \mathbf{S_{xx}}^{-1/2}\frac{\mathbf{Q}}{||\mathbf{Q}||_F} \\
&= \mathbf{S_{xx}}^{-1/2}\frac{1}{||\mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}S_{yy}^{-1/2}||_F}\mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}S_{yy}^{-1/2} \\
&= \frac{1}{||\mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}||_F}\mathbf{S_{xx}}^{-1}\mathbf{S_{xy}} \\
&= c(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}
\end{aligned}$$

where c is a scalar constant.

Thus, we can see that the first canonical correlation vector $\mathbf{a}$ is a scalar multiple of

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y},$$

the classical least squares estimator. Therefore the least squares estimator $\hat{\boldsymbol{\beta}}$ solves (5.3). However, it does not usually solve the constrained optimisation problem (5.6) because typically $\hat{\boldsymbol{\beta}}^\top\mathbf{S_{xx}}\hat{\boldsymbol{\beta}} \neq 1$, so that the constraint in Equation (5.6) will not be satisfied.

5.3.2 Invariance/equivariance properties of CCA

Suppose we apply orthogonal transformations and translations to the $\mathbf{x}_i$ and the $\mathbf{y}_i$ of the form

$$\mathbf{h}_i = \mathbf{T}\mathbf{x}_i + \boldsymbol{\mu} \qquad \text{and} \qquad \mathbf{k}_i = \mathbf{R}\mathbf{y}_i + \boldsymbol{\nu}, \qquad i = 1, \ldots, n, \tag{5.20}$$

where $\mathbf{T}$ $(p \times p)$ and $\mathbf{R}$ $(q \times q)$ are orthogonal matrices, and $\boldsymbol{\mu}$ $(p \times 1)$ and $\boldsymbol{\nu}$ $(q \times 1)$ are fixed vectors.

How do these transformations affect the CC analysis?

Firstly, since CCA depends only on sample covariance matrices, it follows that the translation vectors $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ have no effect on the analysis.

Secondly, let's consider the effect of the rotations by an orthogonal matrix. We've seen that CCA in the original $\mathbf{x}$ and $\mathbf{y}$ coordinates depends on

$$\mathbf{Q} \equiv \mathbf{Q_{xy}} = \mathbf{S_{xx}}^{-1/2}\mathbf{S_{xy}}\mathbf{S_{yy}}^{-1/2}. \tag{5.21}$$

In the new coordinates we have

$$\tilde{\mathbf{S}}_{\mathbf{hh}} = \mathbf{T}\mathbf{S_{xx}}\mathbf{T}^\top, \qquad \tilde{\mathbf{S}}_{\mathbf{kk}} = \mathbf{R}\mathbf{S_{yy}}\mathbf{R}^\top,$$

$$\tilde{\mathbf{S}}_{\mathbf{hk}} = \mathbf{T}\mathbf{S_{xy}}\mathbf{R}^\top = \tilde{\mathbf{S}}_{\mathbf{kh}}^\top,$$

where here and below, a tilde above a symbol is used to indicate that the corresponding term is defined in terms of the new $\mathbf{h}$, $\mathbf{k}$ coordinates, rather than the old $\mathbf{x}$, $\mathbf{y}$ coordinates. Due to the fact that $\mathbf{T}$ and $\mathbf{R}$ are orthogonal,

$$\tilde{\mathbf{S}}_{\mathbf{bh}}^{1/2} = \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{1/2}\mathbf{T}^\top, \qquad \tilde{\mathbf{S}}_{\mathbf{hh}}^{-1/2} = \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{T}^\top$$

$$\tilde{\mathbf{S}}_{\mathbf{kk}}^{1/2} = \mathbf{R}\mathbf{S}_{\mathbf{yy}}^{1/2}\mathbf{R}^\top \qquad \text{and} \qquad \tilde{\mathbf{S}}_{\mathbf{kk}}^{-1/2} = \mathbf{R}\mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{R}^\top.$$

The analogue of (5.21) in the new coordinates is given by

$$\begin{aligned}
\tilde{\mathbf{Q}}_{\mathbf{h}k} &= \tilde{\mathbf{S}}_{\mathbf{hh}}^{-1/2}\tilde{\mathbf{S}}_{\mathbf{h}k}\tilde{\mathbf{S}}_{\mathbf{kk}}^{-1/2} \\
&= \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{T}^\top\mathbf{T}\mathbf{S_{xy}}\mathbf{R}^\top\mathbf{R}\mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{R}^\top \\
&= \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{S_{xy}}\mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{R}^\top \\
&= \mathbf{T}\mathbf{Q_{xy}}\mathbf{R}^\top.
\end{aligned}$$

So, again using the fact that $\mathbf{T}$ and $\mathbf{R}$ are orthogonal matrices, if $\mathbf{Q_{xy}}$ has SVD $\sum_{j=1}^{t}\sigma_j\mathbf{u}_j\mathbf{v}_j^\top$, then $\tilde{\mathbf{Q}}_{\mathbf{hk}}$ has SVD

$$\begin{aligned}
\tilde{\mathbf{Q}}_{\mathbf{hk}} = \mathbf{T}\mathbf{Q_{xy}}\mathbf{R}^\top &= \mathbf{T}\left(\sum_{j=1}^{t}\sigma_j\mathbf{u}_j\mathbf{v}_j^\top\right)\mathbf{R}^\top \\
&= \sum_{j=1}^{t}\sigma_j\mathbf{T}\mathbf{u}_j\mathbf{v}_j^\top\mathbf{R}^\top = \sum_{j=1}^{t}\sigma_j\left(\mathbf{T}\mathbf{u}_j\right)\left(\mathbf{R}\mathbf{v}_j\right)^\top = \sum_{j=1}^{t}\sigma_j\tilde{\mathbf{u}}_j\tilde{\mathbf{v}}_j^\top,
\end{aligned}$$

where, for $j = 1, \ldots, t$, the $\tilde{\mathbf{u}}_j = \mathbf{T}\mathbf{u}_j$ are mutually orthogonal unit vectors, and the $\tilde{\mathbf{v}}_j = \mathbf{R}\mathbf{v}_j$ are also mutually orthogonal unit vectors.

Consequently, $\tilde{\mathbf{Q}}_{\mathbf{h}k}$ has the same singular values as $\mathbf{Q_{xy}}$, namely $\sigma_1, \ldots, \sigma_t$ in both cases, and so the canonical correlation coefficients are invariant with respect

to the transformations (5.20). Moreover, since the optimal linear combinations for the jth CC in the original coordinates are given by $\mathbf{a}_j = \mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{u}_j$ and $\mathbf{b}_j = \mathbf{S}_{\mathbf{yy}}^{-1/2}\mathbf{v}_j$, the optimal linear combinations in the new coordinates are given by

$$\begin{aligned}\tilde{\mathbf{a}}_j &= \mathbf{S}_{\mathbf{hh}}^{-1/2}\mathbf{T}\mathbf{u}_j \\ &= \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{T}^\top\mathbf{T}\mathbf{u}_j \\ &= \mathbf{T}\mathbf{S}_{\mathbf{xx}}^{-1/2}\mathbf{u}_j \\ &= \mathbf{T}\mathbf{a}_j,\end{aligned}$$

and a similar argument shows that $\tilde{\mathbf{b}}_j = \mathbf{R}\mathbf{b}_j$. So under transformations (5.20), the optimal vectors $\mathbf{a}_j$ and $\mathbf{b}_j$ transform in an equivariant manner to $\tilde{\mathbf{a}}_j$ and $\tilde{\mathbf{b}}_j$, respectively, $j = 1, \ldots, t$.

If either of $\mathbf{T}$ or $\mathbf{R}$ in (5.20) is not an orthogonal matrix then the singular values are not invariant and the CC vectors do not transform in an equivariant manner.

5.4 Computer tasks

5.4.0.0.1 Task 1

Consider again the crabs dataset you looked at in the exercises in the chapter on PCA (see 4.4). We now consider a canonical correlation analysis in which one set of variables, the $\mathbf{x}$-set, is given by CL and CW and the other set, the $\mathbf{y}$-set, is given by FL, RW and BD.

```
library(MASS)
?crabs           # read the help page to find out about the dataset
X=as.matrix(crabs[4:8])
n=200                                        # sample size
H=diag(rep(1,n))-rep(1,n)%*%t(rep(1,n))/n  # n times n centering matrix
library(dplyr)
X1 = crabs %>% dplyr::select(CL, CW)  %>%as.matrix
# store CL and CW in X1
Y1 = crabs %>% dplyr::select(FL, RW, BD) %>% as.matrix
# store FL, RW and BD in Y1
Sxx=t(X1)%*%H%*%X1/n                         # find x-variable variance matrix
Syy=t(Y1)%*%H%*%Y1/n                         # find y-variable variance matrix
Sxy=t(X1)%*%H%*%Y1/n                         # find cross-covariance matrix
```

i. calculate $\mathbf{S}_{\mathbf{xx}}^{-1/2}$ and $\mathbf{S}_{\mathbf{yy}}^{-1/2}$ by first computing the spectral decomposition of $\mathbf{S}_{xx}$ and $\mathbf{S}_{yy}$.

ii. Now calculate the matrix $\mathbf{Q}$ and compute its singular value decomposition.

iii. Compute the first pair of CC vectors and CC variables η_1 and ψ_1. What is the 1st canonical correlation?

iv. Plot ψ_1 vs η_1. What does the plot tell you (if anything)?

v. Repeat the above to find the second pair of CC vectors, and the second set of CC variables/scores, and plot these against each other and against the first CC scores. Is there any interesting structure in any of the plots? Which plots suggest random scatter?

vi. Finally, repeat the analysis above using the `cc` command and `plt.cc` from the package `CCA` which you will need to download.

```
cca<-cc(X1,Y1)
plt.cc(cca, var.label=TRUE)
```

5.4.0.0.2 Task 2

The full Premier League dataset is available at https://www.rotowire.com/soccer/league-table.php?season=2018. There is a button to download the csv (comma separated variable) file in the bottom right hand corner.

Read the data into R (hint: try the `read.csv` command).

```
x <- read.csv(x , file="/YOURDIRECTORY/prem_league_data.txt, sep=" ", header=TRUE)
```

If you are not sure what the name of YOURDIRECTORY is where the file is located, then a useful command to find out is `file.choose()`

i. Check that you can reproduce, and agree with, the calculations done in this chapter.

ii. Consider now doing CCA with $\mathbf{x} = (W, D)$ and $\mathbf{y} = (G, GA, L)$. Note that if you knew W and D, you could calculate L. Without doing any computation, what do you expect the first canoncial correlation to be? What will the first pair of CC vectors be (upto a multiplicative constant)?

iii. Check your intuition by doing the calculation in R:

```
X <- table[,c('W','D')]
Y <- table[,c('G','GA','L')]
cc(X,Y)
```

5.4.0.0.3 Task 3

We will now look data measured from 600 first year university students. Measurements were made on three psychological variables:

- Locus of Control: the degree to someone believes that they, as opposed to external forces, have control over the outcome of events in their lives.
- Self Concept: an indication of whether a person tends to hold a generally positive and consistent or negative and variable self-view.
- Motivation: how motivated an individual is

which will form our $\mathbf{X}$ variables. The $\mathbf{Y}$ variables are four academic scores (standardized test scores)

- Reading
- Writing
- Math
- Science

and gender (1=Male, 0 = Female) We are interested in how the set of psychological variables relates to the academic variables and gender.

```
mm <- read.csv("https://stats.idre.ucla.edu/stat/data/mmreg.csv")
colnames(mm) <- c("Control", "Concept", "Motivation", "Read", "Write", "Math",
    "Science", "Sex")
summary(mm)
psych <- mm[, 1:3]
acad <- mm[, 4:7]
```

Conduct CCA on these data. Provide an interpretation of your results.

5.5 Exercises

1. Attempt exam question 1 part (b) from the 2017-18 exam paper. You will find the past exam papers on Moodle.

2. Suppose that $\mathbf{z} = (\mathbf{x}^\top \mathbf{y}^\top)^\top$ is a random vector, where both $\mathbf{x}$ and $\mathbf{y}$ are sub-vectors of dimension p, so that $\mathbf{z}$ is $(2p) \times 1$. Define

$$\mathbb{V}\mathrm{ar}(\mathbf{z}) = \mathbf{\Sigma_{zz}} = \begin{pmatrix} \mathbf{\Sigma_{xx}} & \mathbf{\Sigma_{xy}} \\ \mathbf{\Sigma_{yx}} & \mathbf{\Sigma_{yy}} \end{pmatrix}.$$

   i. Suppose that $\mathbf{y} = \mathbf{T}\mathbf{x}$ where $\mathbf{T}$ is a fixed matrix. Find $\mathbf{\Sigma_{xy}}$ and $\mathbf{\Sigma_{yy}}$ in terms of $\mathbf{\Sigma_{xx}}$ and $\mathbf{T}$.
   ii. Assuming now that $\mathbf{T}$ is an orthogonal matrix and $\mathbf{\Sigma_{xx}}$ is of full rank, determine the singular values of the matrix $\mathbf{Q} = \mathbf{\Sigma_{xx}}^{-1/2}\mathbf{\Sigma_{xy}}\mathbf{\Sigma_{yy}}^{-1/2}$, and hence write down the canonical correlation coefficients.
   iii. Suppose now that $\mathbf{T}$ is non-singular but not orthogonal. Comment on whether the answer to part (b) changes.

3. We will now prove Proposition 5.5 by induction. The case for $k = 1$ was proved in Section 5.1 in Proposition 5.2. Assume the result is true for k. Consider the objective

$$\mathcal{L} = \mathbf{a}^\top \mathbf{Q}\mathbf{b} + \sum_{i=1}^{k} \gamma_i \mathbf{a}^\top \mathbf{a}_i + \sum_{i=1}^{k} \mu_i \mathbf{b}^\top \mathbf{b}_i + \frac{\lambda_1}{2}(1 - \mathbf{a}^\top \mathbf{a}) + \frac{\lambda_2}{2}(1 - \mathbf{b}^\top \mathbf{b})$$

   where $\lambda_i, \mu_i, \gamma_i$ are Lagrangian multipliers.

i. By differentiating with respect to $\mathbf{a}$ and $\mathbf{b}$ and setting the derivative to zero show that

$$\mathbf{Q}\mathbf{b} + \sum \gamma_i \mathbf{a}_i - \lambda_1 \mathbf{a} = 0 \tag{5.22}$$

$$\mathbf{Q}^\top \mathbf{a} + \sum \mu_i \mathbf{b}_i - \lambda_2 \mathbf{b} = 0. \tag{5.23}$$

ii. By left multiplying the equations above by $\mathbf{a}^\top$ and $\mathbf{b}^\top$ respectively show that

$$\lambda_1 = \lambda_2 = \mathbf{a}^\top \mathbf{Q}\mathbf{b}.$$

iii. By left multiplying (5.22) by $\mathbf{a}_i^\top$ show that $\gamma_i = 0$ for $i = 1, \ldots, k$. Show similarly that $\mu_i = 0$ for $i = 1, \ldots, k$.

iv. Finally, by copying the proof of Proposition 5.2, prove Proposition 5.5.

4. Show the mean of the cc variables η_k and ψ_k is zero. Prove Proposition 5.6 giving the variance of covariance the cc variables.

Chapter 6

Multidimensional Scaling (MDS)

The videos for this chapter are available at:

- 6.0 Introduction to MDS
- 6.1 Classical MDS
- 6.1 Example 1
- 6.1.1 Non-Euclidean distance matrices
- 6.1.2 Principal Coordinate Analysis
- 6.2 Similarity measures
- 6.2.1 Binary attributes
- 6.3 Non-metric MDS

In PCA, we start with n data points $\mathbf{x}_i \in \mathbb{R}^p$, and then try to find a low dimensional projection of these points, e.g., $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^r$ with $r < p$, in such a way that they minimize the reconstruction error (or maximize the variance).

The focus in **Multidimensional Scaling (MDS)** is somewhat different. Instead of being given the data $\mathbf{X}$, our starting point is often a matrix of **distances** or **dissimilarities** between the data points, $\mathbf{D}$. For example, if we have data on n different experimental units, then we would be given the distances d_{ij} between any pair of experimental units i and j. We compile these into a $n \times n$ **distance matrix** $\mathbf{D} = (d_{ij} : i, j = 1, \ldots, n)$.

The goal of MDS is to find a set of points in a low-dimensional Euclidean space $\mathbb{R}^r$, usually $\mathbb{R}$ or $\mathbb{R}^2$, whose inter-point distances (or dissimilarities) are as close as possible to the d_{ij}. That is, we want to find $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^r$ whose distance matrix is approximately $\mathbf{D}$, i.e., for which

$$\text{distance}(\mathbf{y}_i, \mathbf{y}_j) \approx d_{ij}.$$

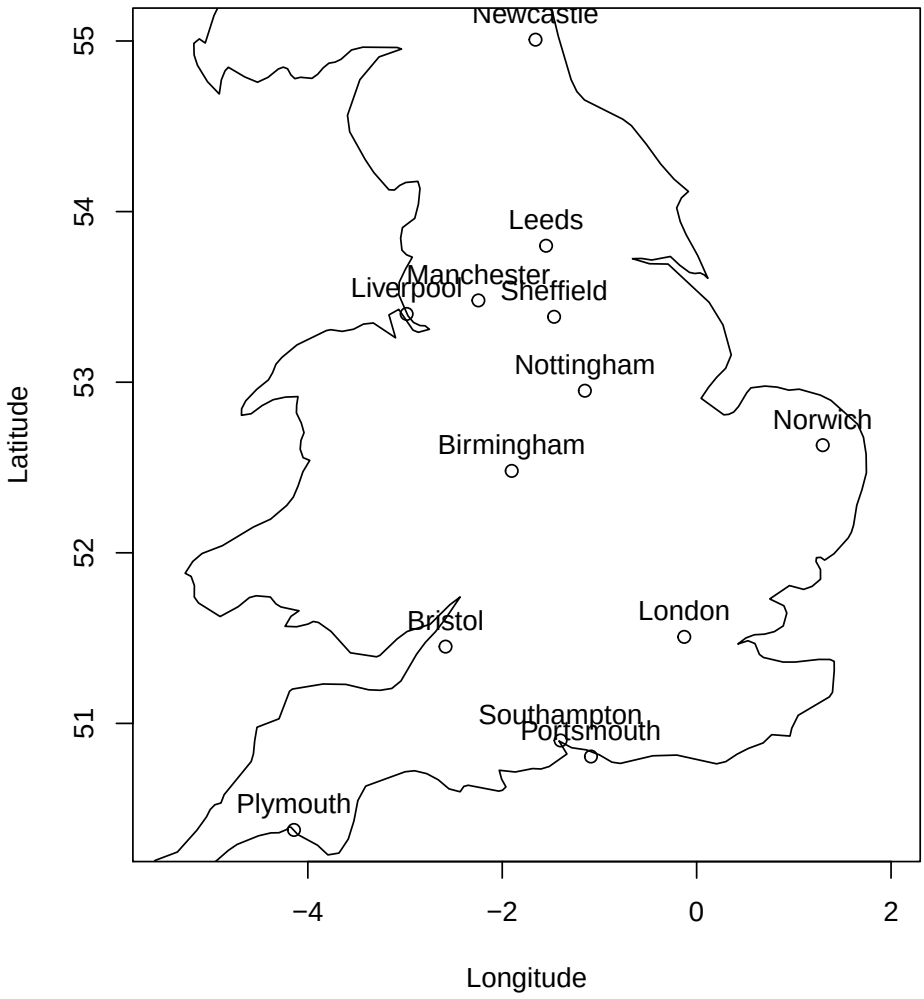
In other words, we are trying to create a spatial representation of the data, $\mathbf{y}_1, \ldots, \mathbf{y}_n$, from a distance matrix $\mathbf{D}$. The vectors $\mathbf{y}_i$ have no meaning by themselves, but by visualising their spatial pattern we can hope to learn something about the dataset represented by $\mathbf{D}$.

If we define the errors in terms of a square distance, then we can write the goal of MDS as the following optimization problem:

$$\text{Find} \quad \mathbf{y}_1, \ldots, \mathbf{y}_k \in \mathbb{R}^r \tag{6.1}$$

$$\text{to minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - d(\mathbf{y}_i, \mathbf{y}_j))^2. \tag{6.2}$$

As an illustrative example, consider the location of some of England's cities.



If we are told their latitude and longitude, it is easy to calculate the distances between the cities. I.e., given the latitude and longitude of each city, $\mathbf{x}_i$, we can compute the distance matrix between pairs of cities $\mathbf{D}$:

	London	Birmingham	Manchester	Leeds	Newcastle	Liverpool	Portsmouth	Southampton	Nottingham	Br
London	0	163	262	273	403	286	103	112	175	
Birmingham	163	0	114	149	282	125	195	179	73	
Manchester	262	114	0	58	174	50	308	293	94	
Leeds	273	149	58	0	135	105	335	323	98	
Newcastle	403	282	174	135	0	199	469	458	231	
Liverpool	286	125	50	105	199	0	317	299	132	
Portsmouth	103	195	308	335	469	317	0	24	239	
Southampton	112	179	293	323	458	299	24	0	229	
Nottingham	175	73	94	98	231	132	239	229	0	
Bristol	170	124	227	271	401	219	127	103	193	
Sheffield	228	105	53	47	181	101	288	276	53	
Norwich	159	217	255	230	328	299	261	268	169	
Plymouth	308	281	369	420	542	346	221	202	353	

But can we do the reverse and construct a map from the distance matrix? This is the aim of multidimensional scaling: MDS constructs a set of points, $\mathbf{y}_1, \ldots, \mathbf{y}_n$, that have distances between them given by the distance matrix $\mathbf{D}$. In other words, it creates a map with a set of coordinates for which the distances between points are approximately the same as in the real data.

Of course, this illustrative example is not very interesting, as the original data (the city locations) are only two-dimensional, but in problems with high dimensional data, finding a way to represent the points in a low-dimensional space will make visualization and statistical analysis easier. We shall see, perhaps unsurprisingly, that there is a close connection between MDS and PCA.

Notation

There are many ways to compute distances between points. For example, you may have studied metrics previously, which are distance functions that satisfy certain axioms. In MDS, we do not need to work with distances that are metrics (although sometimes we do), and instead we only require that distances satisfy a weaker set of conditions:

Definition 6.1. The $n \times n$ matrix $\mathbf{D} = (d_{ij})_{i,j=1}^n$ is a **distance matrix** (sometimes called a **dissimilarity matrix**) if

1. $d_{ij} \geq 0$ for all $i, j = 1, \ldots, n$.
2. $d_{ii} = 0$ for $i = 1, \ldots, n$ and
3. $\mathbf{D} = \mathbf{D}^\top$, i.e., $\mathbf{D}$ is symmetric ($d_{ij} = d_{ji}$).

Note that we do not require distances to necessarily satisfy the triangle inequality

$$d_{ik} \leq d_{ij} + d_{jk}. \tag{6.3}$$

A distance function which always satisfies the triangle inequality is called a **metric distance** or just a **metric**. A distance function which does not always satisfy the triangle inequality is called a **non-metric** distance.

There are many ways to measure distance. Two common choices of metric distances are

- Euclidean distance $||\mathbf{x} - \mathbf{x}'||_2$, also know as the 'crow flies' distance.
- L_1 distance $||\mathbf{x} - \mathbf{x}'||_1$, sometimes called the Manhattan or taxicab metric.

But for the city data, we could also consider distance by road, i.e., we could measure the minimum distance by road between each pair of cities. This would satisfy the axioms for being a distance, but is not a metric distance

The choice of which distance is appropriate is problem specific (see section 6.2 for an example). The focus in this chapter is on Euclidean distance, as this leads to an explicit mathematical solution to the optimization problem (6.2). However, note that MDS techniques have also been developed for non-Euclidean distances.

6.1 Classical MDS

Classical MDS is based upon **Euclidean distances**:

$$d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2 = \sqrt{(\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}')}.$$

Since Euclidean distances satisfy the triangle inequality (6.3), it follows that Euclidean distance is a metric distance. In general, any distance derived from a norm is a metric distance.

Definition 6.2. Suppose $\mathbf{D}$ is an $n \times n$ distance matrix. We say $\mathbf{D}$ is a **Euclidean distance matrix** if there is a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ for some p, such that

$$d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||_2 = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)}.$$

In classical MDS, we don't work directly with the distance matrix, but with the centred inner product matrix.

Definition 6.3. Given a distance matrix $\mathbf{D} = \{d_{ij}\}_{i,j=1}^n$, the **centred inner-product matrix** (also called the **centred-Gram matrix**) is

$$\mathbf{B} = \mathbf{HAH}, \tag{6.4}$$

where $\mathbf{A}$ is the matrix of negative square distances divided by two:

$$\mathbf{A} = \{a_{ij}\}_{i,j=1}^n, \quad \text{where} \qquad a_{ij} = -\frac{1}{2} d_{ij}^2 \tag{6.5}$$

and $\mathbf{H}$ is the $n \times n$ **centering matrix** (see Section 2.4).

Another way of writing $\mathbf{A}$ is as

$$\mathbf{A} = -\frac{1}{2}\mathbf{D} \odot \mathbf{D}$$

where $\odot$ denotes the Hadamard (or element-wise) product of two matrices.

We can think of $\mathbf{B}$ as a matrix of similarities for the data. The reason for this, and for why we call $\mathbf{B}$ the centred inner product matrix will become clear after we state the main result of this chapter.

Main result

We now present the key result for classical MDS. It says that a distance matrix $\mathbf{D}$ is a *Euclidean* distance matrix if and only if the corresponding centred inner product matrix $\mathbf{B}$ is positive semi-definite. It also tells us how given data $\mathbf{X}$ we can compute $\mathbf{B}$ directly from $\mathbf{X}$. Conversely, and more importantly, it tells us how given $\mathbf{B}$ we can compute some data points $\mathbf{X}$ that have corresponding Euclidean distance matrix $\mathbf{D}$.

Theorem 6.1. *Let $\mathbf{D}$ denote an $n \times n$ distance matrix with corresponding centred inner-product matrix $\mathbf{B} = -\frac{1}{2}\mathbf{H}(\mathbf{D} \odot \mathbf{D})\mathbf{H}$.*

1. *If $\mathbf{D}$ is a **Euclidean** distance matrix for the sample of n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, then*

$$\mathbf{B} = (\mathbf{HX})(\mathbf{HX})^\top. \tag{6.6}$$

   Thus $\mathbf{B}$ is positive semi-definite.

2. *Suppose $\mathbf{B}$ is positive semi-definite with eigenvalues $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_k > 0$ and that it has spectral decomposition $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{\Lambda} = diag\{\lambda_1 \ldots \lambda_k\}$ and $\mathbf{U}$ is $n \times k$ and satisfies $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_k$. Then*

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top = \mathbf{U}\mathbf{\Lambda}^{1/2}$$

   *is an $n \times k$ data matrix which has **Euclidean** distance matrix $\mathbf{D}$.*

   Moreover, for this data matrix $\bar{\mathbf{x}} = \mathbf{0}_k$ and $\mathbf{B}$ represents the inner product matrix with elements given by (6.6)*.*

For part 1. we may equivalently write

$$b_{ij} = (\mathbf{x}_i - \bar{\mathbf{x}})^\top(\mathbf{x}_j - \bar{\mathbf{x}}), \qquad i, j = 1, \ldots, n, \tag{6.7}$$

where $\bar{\mathbf{x}} = n^{-1}\sum_{i=1}^n \mathbf{x}_i$ is the sample mean vector. This illustrates why we call $\mathbf{B}$ the *centred inner-product* matrix: $\mathbf{HX}$ is the centred data matrix, and $(\mathbf{HX})(\mathbf{HX})^\top$ thus contains the inner product of each pair of centred vectors.

We think of $\mathbf{B}$ as a matrix of pair-wise **similarities** between the data points. The inner product

$$\langle \mathbf{x}_i - \bar{\mathbf{x}},\ \mathbf{x}_j - \bar{\mathbf{x}} \rangle = (\mathbf{x}_i - \bar{\mathbf{x}})^\top(\mathbf{x}_j - \bar{\mathbf{x}})$$

is large if $\mathbf{x}_i$ is similar to $\mathbf{x}_j$, and small if they are very different. It may help to think back to the geometric interpretation of inner products from Section 2.3.1.

Proof. You will prove this result in the example sheets. □

Corollary 6.1. *The distance matrix* $\mathbf{D}$ *is a Euclidean distance matrix if and only if the corresponding centred inner-product matrix* $\mathbf{B}$ *is a positive semi-definite matrix.*

These results say that if $\mathbf{D}$ is a Euclidean distance matrix (which we can check by testing whether $\mathbf{B}$ is positive semi-definite), then we can find a set of n points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^k$ which have interpoint distances $\mathbf{D}$. The dimension of the points is given by the rank of $\mathbf{B}$.

Example 1

Consider the five point in $\mathbb{R}^2$:

$$\mathbf{x}_1 = (0,0)^\top, \mathbf{x}_2 = (1,0)^\top, \quad \mathbf{x}_3 = (0,1)^\top$$
$$\mathbf{x}_4 = (-1,0)^\top \quad \text{and} \quad \mathbf{x}_5 = (0,-1)^\top.$$

The resulting distance matrix is

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & \sqrt{2} & 2 & \sqrt{2} \\ 1 & \sqrt{2} & 0 & \sqrt{2} & 2 \\ 1 & 2 & \sqrt{2} & 0 & \sqrt{2} \\ 1 & \sqrt{2} & 2 & \sqrt{2} & 0 \end{bmatrix}.$$

The aim of MDS is to construct a set of 5 points in $\mathbb{R}^k$ for some choice of k, that have interpoint distances given by $\mathbf{D}$.

Using (6.5) first to calculate $\mathbf{A}$, and then using (6.4) to calculate $\mathbf{B}$, we find that

$$\mathbf{A} = -\begin{bmatrix} 0 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0 & 1 & 2 & 1 \\ 0.5 & 1 & 0 & 1 & 2 \\ 0.5 & 2 & 1 & 0 & 1 \\ 0.5 & 1 & 2 & 1 & 0 \end{bmatrix}$$

and

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}.$$

In R we can compute these as follows:

```
X <- matrix(c(0,0,
              1,0,
              0,1,
              -1,0,
              0,-1), nc=2, byrow=TRUE)
D <- as.matrix(dist(X, upper=T, diag=T))
A <- -D^2/2
# note D^2 does element wise operations, different to D%*%D
H <- diag(5) - 1/5 * matrix(rep(1,5), nc=1)%*%matrix(rep(1,5), nr=1)
B <- H%*% A%*%H #check this matches  (H %*% X) %*% t(H %*% X)
```

You should check that $\mathbf{HAH}$ is the same as $(\mathbf{HX})(\mathbf{HX})^\top$, thus verifying part 1. of Theorem 6.1. Note that there will be very small differences ($\sim 10^{-16}$) due to numerical errors.

How do we construct, using only the information in $\mathbf{B}$, a set of 5 points that have Euclidean interpoint distances given by $\mathbf{D}$? Firstly, we need to compute the spectral decomposition of $\mathbf{B}$. The eigenvalues of $\mathbf{B}$ are

$$\lambda_1 = \lambda_2 = 2 \qquad \text{and} \qquad \lambda_3 = \lambda_4 = \lambda_5 = 0.$$

Thus $\mathbf{B}$ is positive semi-definite, and so by Theorem 6.1, $\mathbf{D}$ must be a Euclidean distance matrix. This isn't a surprise to us because we created $\mathbf{D}$ ourselves in this case.

There are two positive eigenvalues, and so $\mathbf{B}$ has rank 2, and the theorem tells us we can construct points $\mathbf{x}_i \in \mathbb{R}^2$, i.e., points in 2-dimensional space.

To find the points we need to find the matrix of orthogonal unit eigenvectors of $\mathbf{B}$, which we denoted as $\mathbf{U}$ in the theorem. Because we have a repeated eigenvalue ($\lambda = 2$ has *multiplicity* 2), the eigenspace associated with $\lambda = 2$ is a two dimensional space. There is not a unique pair of orthogonal unit eigenvectors spanning this space (there are an infinite number of possible pairs). The sparsest choice (i.e. the one with most zero elements) is

$$\mathbf{u}_1 = \begin{pmatrix} 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \qquad \text{and} \qquad \mathbf{u}_2 = \begin{pmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}.$$

Note that if we compute these in R, you may well get different eigenvectors, but the subspace described by the pair of vectors will be the same.

```
eigen(B)$values #using B = H%*% A%*%H
```

```
## [1]  2.000000e+00  2.000000e+00  1.207037e-15  3.331224e-16 -2.775558e-17
```

```
eigen(B)$vectors[,1:2]
```

```
##            [,1]       [,2]
## [1,]  0.0000000  0.0000000
## [2,] -0.6916609  0.1469869
## [3,] -0.1469869 -0.6916609
## [4,]  0.6916609 -0.1469869
## [5,]  0.1469869  0.6916609
```

```
B2 <- (H %*% X) %*% t(H %*% X) # alternative way of computing B
eigen(B2)$values
```

```
## [1] 2.000000e+00 2.000000e+00 1.110223e-15 1.110223e-15 0.000000e+00
```

```
eigen(B2)$vectors[,1:2]
```

```
##            [,1]       [,2]
## [1,]  0.0000000  0.0000000
## [2,]  0.0000000 -0.7071068
## [3,] -0.7071068  0.0000000
## [4,]  0.0000000  0.7071068
## [5,]  0.7071068  0.0000000
```

We can now compute the coordinates of five points in $\mathbb{R}^2$ which have Euclidean distance matrix, $\mathbf{D}$, by

$$\mathbf{U}\mathbf{\Lambda}^{1/2} = \sqrt{2}[\mathbf{u}_1, \mathbf{u}_2].$$

Note again that these coordinates are not unique as any rotation or translation of them will have the same distance matrix. In particular, when doing computations in R we may find different answers depending upon how we do the computation. So for example, if we use the eigendecompostion of $\mathbf{B} = \mathbf{HAH}$

```
B.eig <- eigen(B)
Y <-sqrt(B.eig$values[1:2])*B.eig$vectors[,1:2]
```

we find

$$\begin{pmatrix} 0 & 0 \\ -0.978 & 0.208 \\ -0.208 & -0.978 \\ 0.978 & -0.208 \\ 0.208 & 0.978 \end{pmatrix}.$$

Whereas if we use the eigen decomposition of $(\mathbf{HX})(\mathbf{HX})^\top$ we find a different set of points:

```
B2.eig <- eigen(B2)
Y2 <-sqrt(B2.eig$values[1:2])*B2.eig$vectors[,1:2]
```

$$\begin{pmatrix} 0 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The distance matrices for both sets of points are equal to $\mathbf{D}$

```
signif(dist(Y, upper=T, diag=T),2) # should agree with D.
```

```
##     1   2   3   4   5
## 1 0.0 1.0 1.0 1.0 1.0
## 2 1.0 0.0 1.4 2.0 1.4
## 3 1.0 1.4 0.0 1.4 2.0
## 4 1.0 2.0 1.4 0.0 1.4
## 5 1.0 1.4 2.0 1.4 0.0
```

```
signif(dist(Y2, upper=T, diag=T),2) # should agree with D.
```

```
##     1   2   3   4   5
## 1 0.0 1.0 1.0 1.0 1.0
## 2 1.0 0.0 1.4 2.0 1.4
## 3 1.0 1.4 0.0 1.4 2.0
## 4 1.0 2.0 1.4 0.0 1.4
## 5 1.0 1.4 2.0 1.4 0.0
```

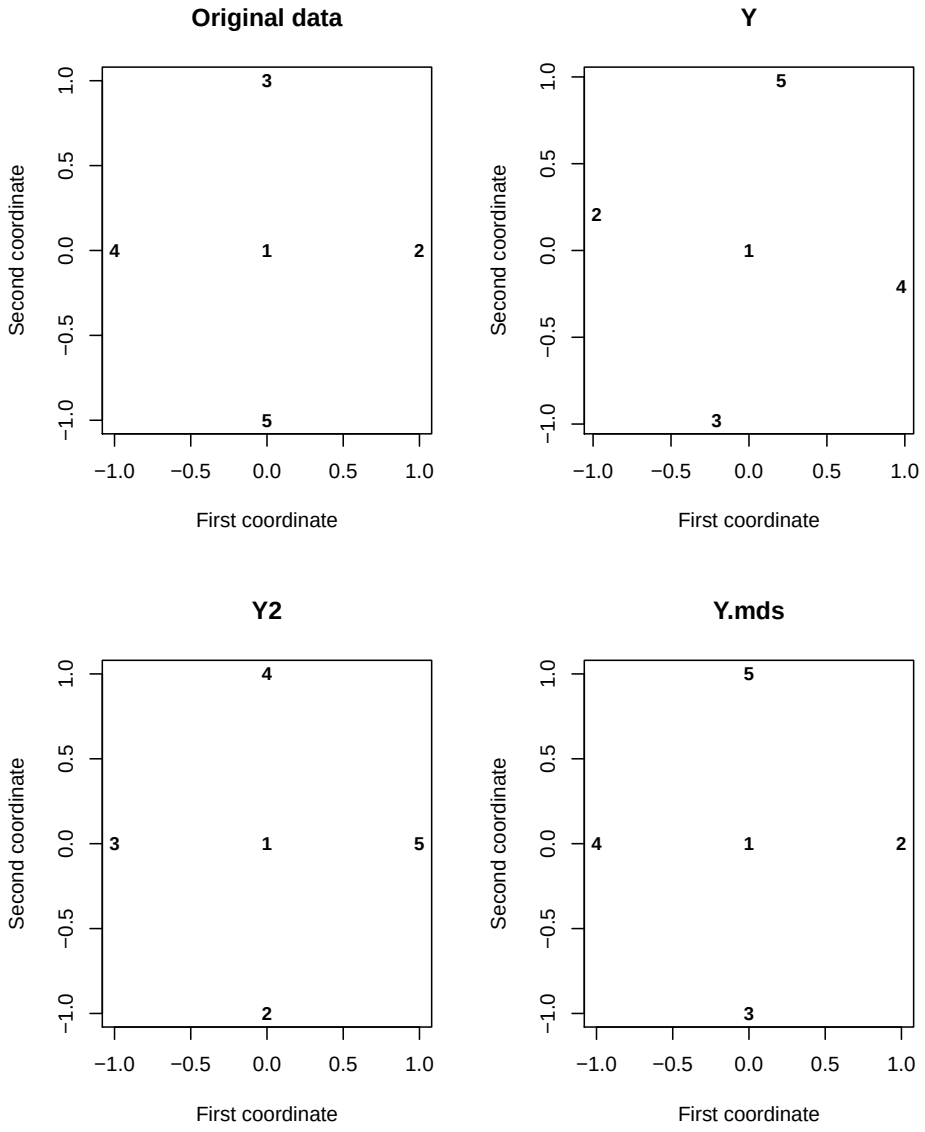
Finally, note that as always, there is an R command that will do all of this work for us

```
Y.mds <- cmdscale(D, eig=TRUE) #Classical MultiDimensional Scaling - cmdscale
```

giving the set of points

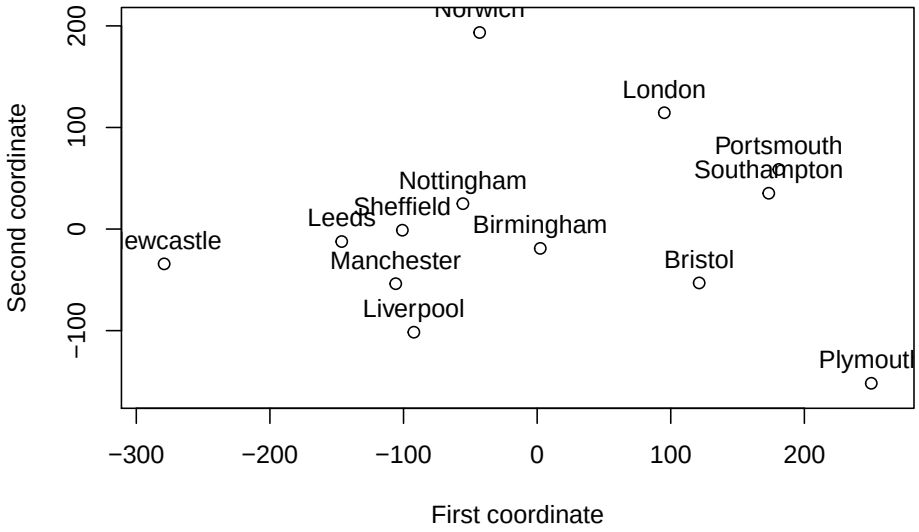
$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

If we plot the original data points $\mathbf{X}$, along with the three sets of reconstructed data points `Y`, `Y2`, and `Y.mds` we can see that they are essentially the same, apart from being rotatated or reordered.



Example 2

If we apply the `cmdscale` to the distances between UK cities from the introduction, we get the 'coordinates' shown below.



This 'map' is essentially correct, but it is rotated 90° and the 'coordinates' have mean zero.

6.1.1 Non-Euclidean distance matrices

Proposition 6.1 may be useful even if $\mathbf{D}$ is not a *Euclidean* distance matrix (in which case $\mathbf{B}$ will have some negative eigenvalues). For example, this can occur if the distances between points is measured with errror.

In this case, instead of using $\mathbf{B}$ in Proposition 6.1 we can use its positive part. If $\mathbf{B}$ has spectral decomposition $\sum_{j=1}^{p} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top$, then its positive definite part is defined by

$$\mathbf{B}_{\text{pos}} = \sum_{j:\, \lambda_j > 0} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top .$$

In other words, we sum over those j such that λ_j is positive.

Then $\mathbf{B}_{\text{pos}}$ is positive semi-definite and so we can use part 2. of Theorem 6.1 to determine a Euclidean configuration of points which has centred inner-product matrix $\mathbf{B}_{\text{pos}}$. Then, provided the negative eigenvalues are small in absolute value relative to the positive eigenvalues, the inter-point distances of the new points in Euclidean space should provide a good approximation to the original inter-point distances (d_{ij}).

Example 3

Let's now look at a case for which $\mathbf{B}$ isn't positive definite. We'll create this by modifying the distance matrix we had before:

$$\mathbf{D}_2 = \begin{pmatrix} 0 & 0.5 & 1 & 1 & 1 \\ 0.5 & 0 & 1.41 & 2 & 1.41 \\ 1 & 1.41 & 0 & 1.41 & 2 \\ 1 & 2 & 1.41 & 0 & 1.41 \\ 1 & 1.41 & 2 & 1.41 & 0 \end{pmatrix}.$$

This is a distance matrix, but is it a *Euclidean* distance matrix? I.e., is there a set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_5$ which have Euclidean distances between them given by $\mathbf{D}_2$?

If we look at the eigenvalues of the centred inner-product matrix $\mathbf{B}$ associated with $\mathbf{D}_2$, then we find that it has negative eigenvalues, and is thus not positive semi-definite. Thus by Theorem 6.1 $\mathbf{D}_2$ is not a *Euclidean* distance matrix.

```
D2 <- D
D2[2,1]<-0.5
D2[1,2]<-D2[2,1]
A2 <- -D2^2/2
B2 <- H%*% A2%*%H
eigen(B2)$values
```

```
## [1]  2.026016e+00  2.000000e+00  1.004310e-01 -7.216450e-16 -2.764470e-01
```

We can still use classical multidimensional scaling to find a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_5$ that have distances approximately given by $\mathbf{D}_2$. `cmdscale` allows us to specify the dimension of the points $\mathbf{x}_i$. If we pick $\dim(\mathbf{x}) = 2$ then it gives us the set of points

```
cmdscale(D2, k=2)
```

```
##           [,1]          [,2]
## 1 -0.13881300  1.449541e-15
## 2 -0.97216111  1.011091e-14
## 3  0.04112656 -1.000000e+00
## 4  1.02872100 -1.018790e-14
## 5  0.04112656  1.000000e+00
```

which have a distance matrix that approximates $\mathbf{D}_2$, but does not equal it. By using a different number of dimensions, we may be able to get a set of points that have a distance matrix closer to $\mathbf{D}_2$, but we will not be able to find a set of 5 points that have a distance matrix exactly equal to $\mathbf{D}_2$.

To do this ourselves (i.e., not using `cmdscale`) we can use the commands

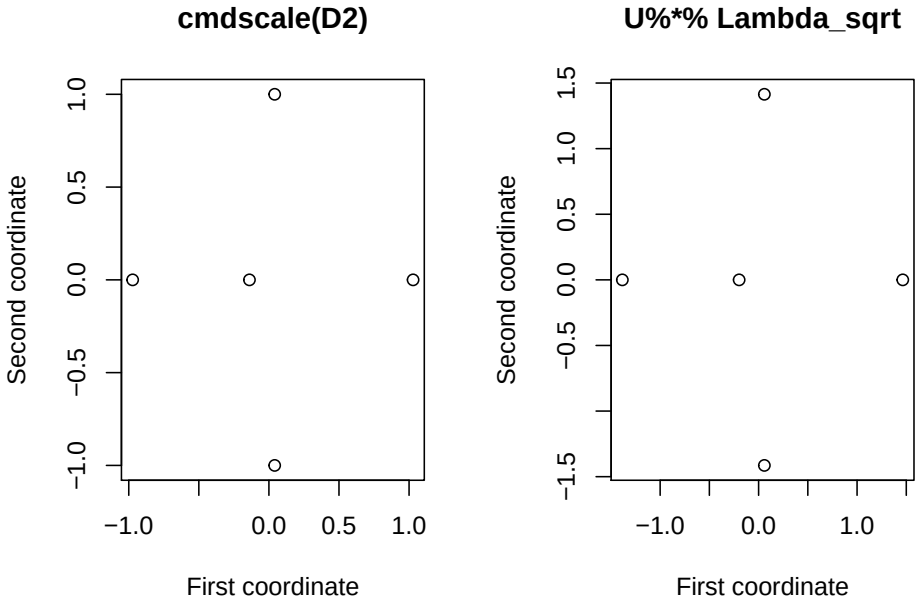
```
U <- eigen(B2)$vectors[,1:2]
Lambda_sqrt <- diag(sqrt(eigen(B2)$values[1:2]))
Y <- U %*% Lambda_sqrt
dist(Y)
```

```
##           1         2         3         4
```

```
## 2 0.8333481
## 3 1.0160602 1.4236404
## 4 1.1675340 2.0008821 1.4054689
## 5 1.0160602 1.4236404 2.0000000 1.4054689
```

```
dist(cmdscale(D2,k=2))
```

```
##           1         2         3         4
## 2 0.8333481
## 3 1.0160602 1.4236404
## 4 1.1675340 2.0008821 1.4054689
## 5 1.0160602 1.4236404 2.0000000 1.4054689
```



6.1.2 Principal Coordinate Analysis

Theorem 6.1 tells us that if the centred inner-product matrix $\mathbf{B}$ is positive semi-definite with rank k, then we can find a set of n points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^k$ that have distance matrix $\mathbf{D}$. However, if k is large this may not be much use to us. The most common motivation for doing MDS is in order to find a way to visualise a dataset and to understand its structure. Thus, we are usually only interested in finding a set of points in $\mathbb{R}^2$ or $\mathbb{R}^3$ that have a distance matrix approximately equal to $\mathbf{D}$ so that we can visualize the points and look for patterns.

Classical MDS (sometimes also called Principal Coordinate Analysis (PCoA)) tries to find a set of points that has approximately the same inner-product matrix, i.e., the same similarities, as the original data, but typically with the points in a low dimensional space. So we want to find $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{R}^r$ (where typically $r = 2$) which have inner-product close to $\mathbf{B}$.

As we are choosing the points $\mathbf{z}_i$, we can choose them so that they have sample mean of $\mathbf{0}$. Then the coresponding inner product matrix is

$$\mathbf{B}_\mathbf{z} = \mathbf{Z}\mathbf{Z}^\top \qquad \text{where} \qquad \mathbf{Z} = \begin{pmatrix} - & \mathbf{z}_1^\top & - \\ - & \mathbf{z}_2^\top & - \\ \vdots & \vdots & \vdots \\ - & \mathbf{z}_n^\top & - \end{pmatrix}.$$

We want to choose $\mathbf{Z}$ so that $\mathbf{Z}\mathbf{Z}^\top \approx \mathbf{B}$. One way to judge how close $\mathbf{B}$ is to $\mathbf{Z}\mathbf{Z}^\top$ is to looking at the sum of square differences between the elements, which is the matrix Frobenius norm:

$$\sum_{i=1}^n \sum_{j=1}^n (B_{ij} - \mathbf{z}_i^\top \mathbf{z}_j)^2 = ||\mathbf{B} - \mathbf{Z}\mathbf{Z}^\top||_F^2$$

Note that if $\mathbf{Z}$ is $n \times r$ then $\mathbf{Z}\mathbf{Z}^\top$ is a rank r matrix. If $\mathbf{B}$, which is a $n \times n$ symmetric matrix, has spectral decomposition $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, then we know from the Eckart-Young-Mirsky Theorem (3.1) that the best rank r approximation to $\mathbf{B}$ (in terms of the Frobenius norm $||\cdot||_F$) is

$$\mathbf{B}_r = \mathbf{U}_r \mathbf{\Lambda}_r \mathbf{U}_r^\top$$

where

$$\mathbf{U}_r = \begin{pmatrix} | & \dots & | \\ \mathbf{u}_1 & \dots & \mathbf{u}_r \\ | & \dots & | \end{pmatrix} \quad \text{and} \quad \mathbf{\Lambda}_r = \text{diag}(\lambda_1, \dots, \lambda_r)$$

are the truncated eigenvector and eigenvalue matrices. If we let

$$\mathbf{Z} = \mathbf{U}_r \mathbf{\Lambda}_r^{\frac{1}{2}}$$

then this gives us the set of n points in $\mathbb{R}^r$ that give the best approximation to the inner product matrix $\mathbf{B}$. These are known as the r leading **principal coordinates** of the data. We will see an example of this in the next section.

Link with PCA

Although MDS only requires either a distance matrix $\mathbf{D}$ or similarity matrix as input, sometimes we will have access to the data matrix $\mathbf{X}$ and will want to reduce the dimension and find a set of points in $\mathbb{R}^2$ that have approximately the same inter-point distances as $\mathbf{X}$. In this case, it is easy to see that classical MDS is the same as PCA.

In PCA, we find the SVD of $\mathbf{H}\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and then if we want a set of points in r dimensions, we use the first r principal component scores

$$\mathbf{Y}_r = \mathbf{U}_r \mathbf{\Sigma}_r.$$

In MDS, we work with the centred Gram matrix $\mathbf{B} = \mathbf{H}\mathbf{X}\mathbf{X}^\top\mathbf{H}$, which has spectral decomposition

$$\mathbf{H}\mathbf{X}\mathbf{X}^\top\mathbf{H} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$$

where $\mathbf{\Sigma}^2 = \mathbf{\Lambda}$ is the diagonal matrix of eigenvalues of $\mathbf{B}$. Thus we can see that the principal coordinates are

$$\mathbf{Z} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}} = \mathbf{U}\mathbf{\Sigma} = \mathbf{Y}.$$

Proposition 6.1. *Let $\mathbf{X}$ be an $n \times p$ data matrix with associated Euclidean distance matrix*

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j),$$

where $\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top$ are the rows of $\mathbf{X}$. Then the centred PC scores based on the first k principal components of $\mathbf{X}$ are equal to the principal coordinates of the n points in k dimensions based on the distance matrix $\mathbf{D}$.

6.2 Similarity measures

So far we have presented classical MDS as starting with a distance (or dissimilarity) matrix $\mathbf{D} = (d_{ij})_{i,j=1}^n$. In this setting, the larger d_{ij} is, the more distant, or dissimilar, object i is from object j. We then convert $\mathbf{D}$ to a centred inner product matrix $\mathbf{B}$, where we think of $\mathbf{B}$ as being a similarity matrix. Finally we find a truncated spectral decomposition for $\mathbf{B}$:

$$\mathbf{D} \longrightarrow \mathbf{B} \longrightarrow \mathbf{Z} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}.$$

Rather than using similarity matrix $\mathbf{B}$ derived from $\mathbf{D}$, we can use a more general concept of **similarity** in MDS.

Definition 6.4. A **similarity matrix** is defined to be an $n \times n$ matrix $\mathbf{F} = (f_{ij})_{i,j=1}^n$ with the following properties:

1. Symmetry, i.e. $f_{ij} = f_{ji}$, $i, j = 1, \ldots, n$.
2. $f_{ij} \leq f_{ii}$ for all $i, j = 1, \ldots, n$.

Note that when working with similarities f_{ij}, the larger f_{ij} is, the more similar objects i and j are.

- Condition 1. implies that object i is as similar to object j as object j is to object i (symmetry).
- Condition 2. implies that an object is at least as similar to itself as it is to any other object.

One common choice of similarity between two vectors is the **cosine** similarity, which is defined to be the cosine of the angle between the vectors. Equivalently, it is the inner product of the vectors after they have been normalised to have length one:

$$\cos\theta = \frac{\mathbf{x}^\top \mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2} = \langle \frac{\mathbf{x}}{||\mathbf{x}||_2}, \frac{\mathbf{y}}{||\mathbf{y}||_2} \rangle$$

This is symmetric (so satisfies property 1. for similarities), and the similarity of a vector $\mathbf{x}$ with itself is 1, and thus property 2. is also satisfied (as $\cos\theta \leq 1$).

Note that because the cosine similarity uses the inner product of normalised vectors, it only gives a relative comparison of two vectors, not an absolute one: the cosine similarity between $\mathbf{x}$ and $\mathbf{y}$ is the same as the similarity between $\mathbf{x}$ and $a\mathbf{y}$ for any positive constant a. Thus we should only use this measure of similarity when absolute differences are unimportant.

MDS with similarity matrices

In this section, we consider MDS using measures of **similarity** as opposed to measures of distance/dissimilarity. We begin by showing that we can convert a positive semi-definite similarity matrix $\mathbf{F}$ into a distance matrix $\mathbf{D}$ and then into a centred inner product matrix $\mathbf{B}$, allowing us to use the classical MDS approach from the previous section.

Proposition 6.2. *Suppose that* $\mathbf{F}$ *is a positive semi-definite similarity matrix. Then the matrix* $\mathbf{D}$ *with elements*

$$d_{ij} = (f_{ii} + f_{jj} - 2f_{ij})^{1/2}, \qquad i, j = 1, \ldots, n \tag{6.8}$$

is a Euclidean distance matrix. Its centred inner product matrix, $\mathbf{B} = -\frac{1}{2}\mathbf{H}(\mathbf{D} \odot \mathbf{D})\mathbf{H}$*, can be computed via*

$$\mathbf{B} = \mathbf{HFH}. \tag{6.9}$$

Proof. Firstly, note that as $\mathbf{F}$ is a similarity matrix, $f_{ii} + f_{jj} - 2f_{ij} \geq 0$ by condition 2., and so the d_{ij} are all well-defined (i.e. real, not imaginary). It is easy to see that $\mathbf{D}$ is a distance matrix as defined in Definition 6.1.

We will now show that Equation (6.9) holds. Let $\mathbf{A} = -\frac{1}{2}\mathbf{D} \odot \mathbf{D}$ as in Equation (6.5). Then

$$a_{ij} = -\frac{1}{2}d_{ij}^2 = f_{ij} - \frac{1}{2}(f_{ii} + f_{jj}).$$

Define

$$t = n^{-1}\sum_{i=1}^{n} f_{ii}.$$

Then, summing over $j = 1, \ldots, n$ for fixed i,

$$\bar{a}_{i+} = n^{-1}\sum_{j=1}^{n} a_{ij} = \bar{f}_{i+} - \frac{1}{2}(f_{ii} + t);$$

similarly,

$$\bar{a}_{+j} = n^{-1} \sum_{i=1}^{n} a_{ij} = \bar{f}_{+j} - \frac{1}{2}(f_{jj} + t),$$

and also

$$\bar{a}_{++} = n^{-2} \sum_{i,j=1}^{n} a_{ij} = \bar{f}_{++} - \frac{1}{2}(t + t).$$

Recall property (vii) from Section 2.4:

$$b_{ij} = a_{ij} - \bar{a}_{i+} - \bar{a}_{+j} + \bar{a}_{++}$$

noting that $\mathbf{A}$ is symmetric. Thus

$$\begin{aligned} b_{ij} &= f_{ij} - \frac{1}{2}(f_{ii} + f_{jj}) - \bar{f}_{i+} + \frac{1}{2}(f_{ii} + t) \\ &\quad - \bar{f}_{+j} + \frac{1}{2}(f_{jj} + t) + \bar{f}_{++} - t \\ &= \quad f_{ij} - \bar{f}_{i+} - \bar{f}_{+j} + \bar{f}_{++}. \end{aligned}$$

Consequently, $\mathbf{B} = \mathbf{HFH}$, again using property (vii) from Section 2.4.

So we've shown that $\mathbf{B} = \mathbf{HFH}$. It only remains to show $\mathbf{D}$ is Euclidean. Since $\mathbf{F}$ is positive semi-definite by assumption, and $\mathbf{H}^\top = \mathbf{H}$, it follows that $\mathbf{B} = \mathbf{HFH}$ must also be positive semi-definite. So by Theorem 6.1 $\mathbf{D}$ is a Euclidean distance matrix. □

This tells us how to do MDS with a similarity matrix $\mathbf{F}$. We first apply double centering to the matrix to get

$$\mathbf{B} = \mathbf{HFH}$$

and then we find the spectral decomposition of $\mathbf{B}$ just as we did in the previous section.

6.2.1 Binary attributes

One important class of problems is when the similarity between any two objects is measured by the number of common attributes. The underlying data on each object is a binary vector of 0s and 1s indicating absence or presence of an attribute. These binary vectors are then converted to similarities by comparing which attributes two objects have in common.

We illustrate this through two examples.

Example 4

Suppose there are 4 attributes we wish to consider.

1. Attribute 1: Carnivore? If yes, put $x_1 = 1$; if no, put $x_1 = 0$.
2. Attribute 2: Mammal? If yes, put $x_2 = 1$; if no, put $x_2 = 0$.
3. Attribute 3: Natural habitat in Africa? If yes, put $x_3 = 1$; if no, put $x_3 = 0$.
4. Attribute 4: Can climb trees? If yes, put $x_4 = 1$; if no, put $x_4 = 0$.

Consider a lion. Each of the attributes is present so $x_1 = x_2 = x_3 = x_4 = 1$. Its attribute vector is $\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}^\top$.

What about a tiger? In this case, 3 of the attributes are present (1, 2 and 4) but 3 is absent. So for a tiger, $x_1 = x_2 = x_4 = 1$ and $x_3 = 0$ or in vector form, its attributes are $\begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}^\top$.

How might we measure the similarity of lions and tigers based on the presence or absence of these four attributes?

First form a 2×2 table as follows.

		1	0
1		a	b
0		c	d

Here a counts the number of attributes common to both lion and tiger; b counts the number of attributes the lion has but the tiger does not have; c counts the number of attributes the tigher has that the lion does not have; and d counts the number of attributes which neither the lion nor the tiger has. In the above, $a = 3$, $b = 1$ and $c = d = 0$.

How might we make use of the information in the 2×2 table to construct a measure of similarity? There are two commonly used measures of similarity.

The **simple matching coefficient (SMC)** counts mutual absence or presence and compares it to the total number of attributes:

$$\frac{a+d}{a+b+c+d}. \tag{6.10}$$

It has a value of 0.75 for this example.

The **Jaccard similarity coefficient** only counts mutual presence and compares this to the number of attributes that are present in at least one of the two objects:

$$\frac{a}{a+b+c}.$$

This is also 0.75 in this example.

Although the Jaccard index and SMC are the same in this case, this is not true in general. The difference between the two similarities can matter. For example,

consider a market basket analysis where we compare shoppers. If a shop sells p different products, we might record the products purchased by each shopper (their 'basket') as a vector of length p, with a 1 in position i if the shopper purchased object i, and 0 otherewise.

The basket of most shoppers might only contain a small fraction of all the available products (i.e. mostly 0s in the attribute vector). In this case the SMC will usually be high when comparing any two shoppers, even when their baskets are very different, as the attribute vectors are mostly 0s. In this case, the Jaccard index will be a more appropriate measure of similarity as it only looks at the difference between shoppers on the basis of the goods in their combined baskets. For example, consider a shop with 100 products and two customers. If customer A bought bread and beer and customer B bought peanuts and beer, then the Jaccard similarity coefficient is $1/3$, but the SMC is $(1 + 97)/100 = 0.98$.

In situations where 0 and 1 carry equivalent information with greater balance across the two groups, the SMC may be a better measure of similarity. For example, vectors of demographic variables stored in dummy variables, such as gender, would be better compared with the SMC than with the Jaccard index since the impact of gender on similarity should be equal, independently of whether male is defined as a 0 and female as a 1 or the other way around.

There are many other possible ways of measuring similarity. For example, we could consider weighted versions of the above if we wish to weight different attributes differently.

Example 5

Let us now consider a similar but more complex example with 6 unspecified attributes (not the same attributes as in Example 1) and 5 types of living creature, with the following data matrix, consisting of zeros and ones.

	1	2	3	4	5	6
$Lion$	1	1	0	0	1	1
$Giraffe$	1	1	1	0	0	1
Cow	1	0	0	1	0	1
$Sheep$	0	0	0	1	0	1
$Human$	0	0	0	0	1	0

Suppose we decide to use the simple matching coefficient (6.10) to measure similarity. Then the following similarity matrix is obtained.

$\mathbf{F} =$

	Lion	Giraffe	Cow	Sheep	Human
$Lion$	1	2/3	1/2	1/2	1/2
$Giraffe$	2/3	1	1/2	1/2	1/6
Cow	1/2	1/2	1	1	1/3
$Sheep$	1/2	1/2	1	1	1/3
$Human$	1/2	1/6	1/3	1/3	1

It is easily checked from the definition that $\mathbf{F} = (f_{ij})_{i,j=1}^{5}$ is a similarity matrix.

Let's see how we would do this in R.

```
animal <- matrix(c(1,1,0,0,1,1,1,1,1,0,0,1,1,0,0,1,0,1,0,0,0,1,0,1,0,0,0,0,1,0),
                 nr=5, byrow=TRUE)
rownames(animal) <- c("Lion", "Giraffe", "Cow", "Sheep", "Human")
colnames(animal)<-paste("A", 1:6, sep="")

SMC <- function(x,y){
  sum(x==y)/length(x)
}
# SMC(animal[1,], animal[2,]) # check this gives what you expect

n=dim(animal)[1]
F_SMC = outer(1:n,1:n,
              Vectorize(function(i,j){
                SMC(animal[i,], animal[j,])
                }
                ))
# we could do this as a double loop, but I've used the outer function
# here.
```

We can use the `dist` function in R to compute this more easily. The `dist` function requires us to specify which metric to use. Here, we use the L_1 distance, which is also known as the Manhattan distance. We have to subtract this from the largest possible distance, which is 6 in this case, to get the similarity, and then divide by 6 to get the SMC.

```
(6-as.matrix(dist(animal, method="manhattan", diag = TRUE, upper = TRUE)))/6
```

```
##              Lion   Giraffe       Cow     Sheep     Human
## Lion    1.0000000 0.6666667 0.5000000 0.3333333 0.5000000
## Giraffe 0.6666667 1.0000000 0.5000000 0.3333333 0.1666667
## Cow     0.5000000 0.5000000 1.0000000 0.8333333 0.3333333
## Sheep   0.3333333 0.3333333 0.8333333 1.0000000 0.5000000
## Human   0.5000000 0.1666667 0.3333333 0.5000000 1.0000000
```

The Jaccard index can be computed as follows:

```
jaccard = function(x, y) {
  bt = table(y, x)
  return((bt[2, 2])/(bt[1, 2] + bt[2, 1] + bt[2, 2]))
}

# jaccard(animal[1,], animal[2,]) # check this gives what you expect
F_jaccard = outer(1:n,1:n, Vectorize(function(i,j){
  jaccard(animal[i,], animal[j,])
  }
```

```
  ))
```

Again, we can compute this using `dist`, but this time using the binary distance metric. See the help page `?dist` to understand why.

```
1-as.matrix(dist(animal, method="binary", diag=TRUE, upper=TRUE))
```

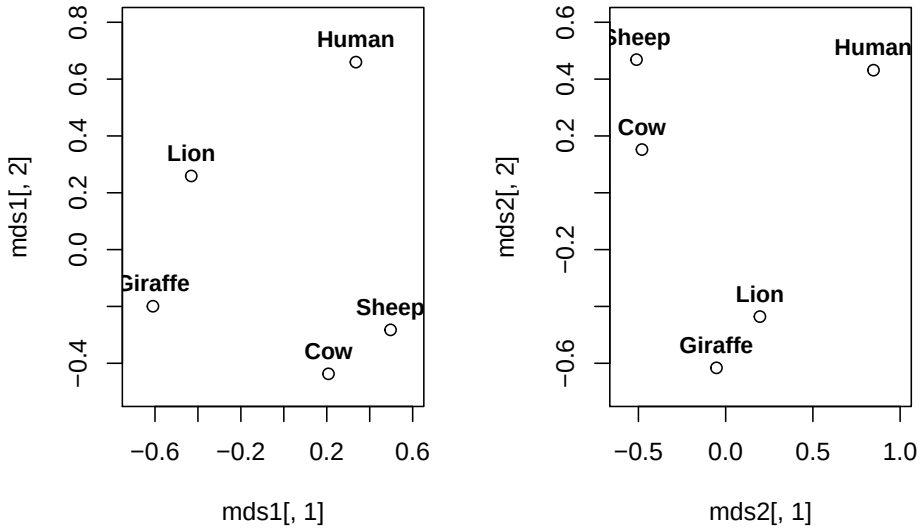
```
##         Lion Giraffe       Cow     Sheep Human
## Lion    1.00     0.6 0.4000000 0.2000000  0.25
## Giraffe 0.60     1.0 0.4000000 0.2000000  0.00
## Cow     0.40     0.4 1.0000000 0.6666667  0.00
## Sheep   0.20     0.2 0.6666667 1.0000000  0.00
## Human   0.25     0.0 0.0000000 0.0000000  1.00
```

To do MDS on these data, we need to first convert from a similarity matrix F to a distance matrix D. We can use the following function to do this:

```
FtoD <- function(FF){
  n = dim(FF)[1]
  D <- matrix(nr=n,nc=n)
  for(ii in 1:n){
    for(jj in 1:n){
      D[ii,jj] <- sqrt(FF[ii,ii]+FF[jj,jj]-2*FF[ii,jj])
    }
  }
  return(D)
}
```

Let's now do MDS, and compare the results from using the SMC and Jaccard index. We could do this by computing $\mathbf{B} = \mathbf{HFH}$ and finding its spectral decomposition, but we will let R do the work for us, and use the `cmdscale` command which takes a distance matrix as input. So let's write a function to convert from a similarity matrix to a distance matrix.

```
mds1<-cmdscale(FtoD(F_SMC))
mds2<-cmdscale(FtoD(F_jaccard))
```



So we can see the choice of index has made a difference to the results.

6.2.2 Example: Classical MDS with the MNIST data

In section 4.3.1 we saw the results of doing PCA on the MNIST handwritten digits. In the final part of that section, we did PCA on a selection of all the digits, and plotted the two leading PC scores, coloured by which digit they represented.

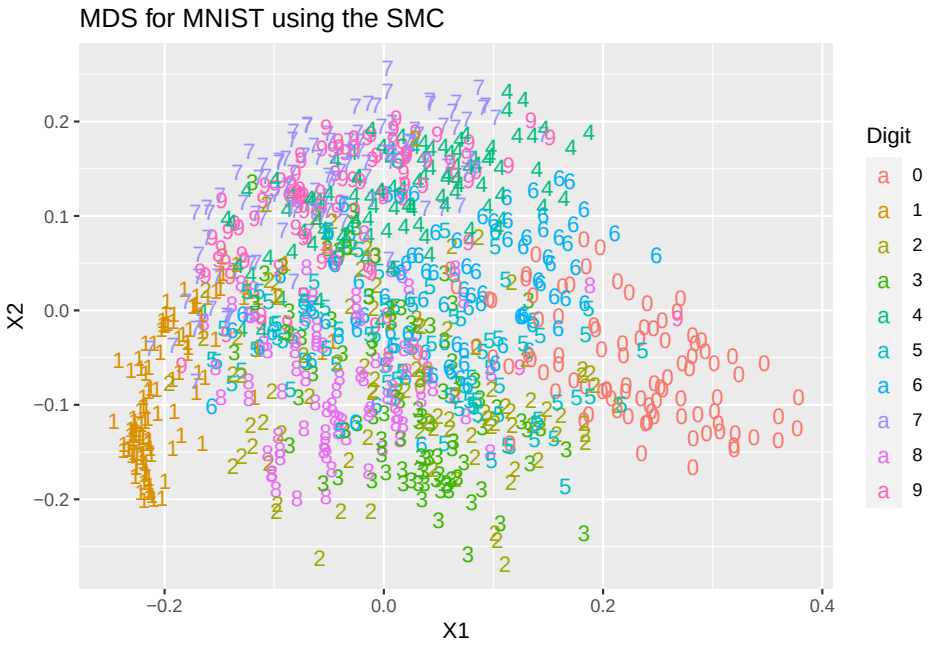
Let's now do MDS on the same data. We know that if we use the Euclidean distance between the image vectors, then we will be doing the same thing as PCA. So let's instead first convert each pixel in the image to binary, with a value of 0 if the intensity is less than 0.3, and 1 otherwise. We can then compute a similarity matrix using the SMC and Jaccard indices.

```
load('mnist.rda')
X<- mnist$train$x[1:1000,]

Y<- (X>0.3)*1.  # multiply by 1 to convert from T/F to a 1/0

n=dim(Y)[1]
p=dim(Y)[2]

F_SMC=(p-as.matrix(dist(Y, method="manhattan", diag = TRUE, upper = TRUE)))/p
F_Jaccard = 1-as.matrix(dist(Y, method="binary", diag=TRUE, upper=TRUE))

mds1=data.frame(cmdscale(FtoD(F_SMC)))
mds2=data.frame(cmdscale(FtoD(F_Jaccard)))
```

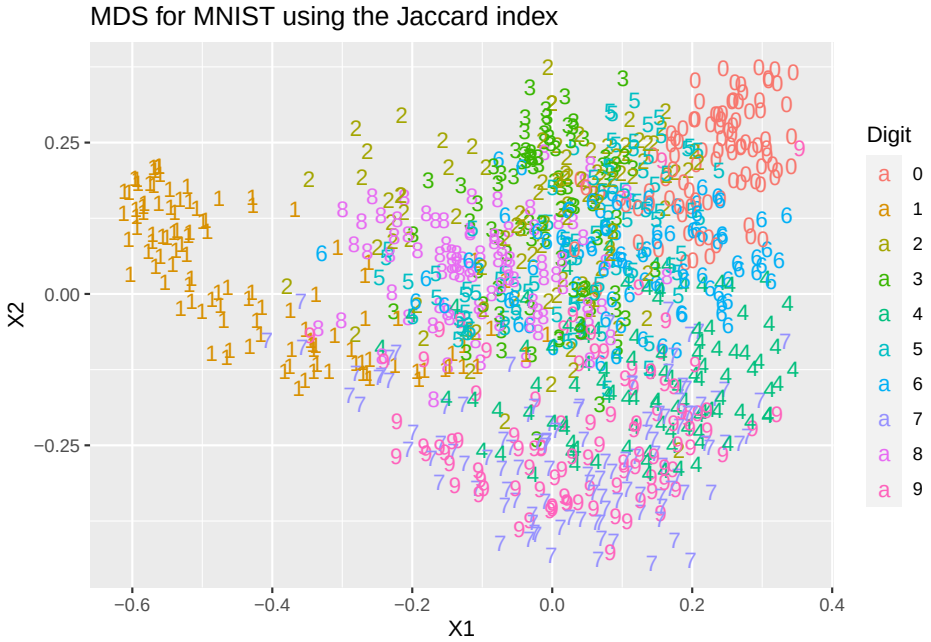
We will do as we did before, and plot the coordinates coloured by the digit each

point is supposed to represent. Note that we have not used these digit labels at any point.

```
Digit = as.factor(mnist$train$y[1:1000])
library(ggplot2)
ggplot(mds1, aes(x=X1, y=X2, colour=Digit, label=Digit))+
  geom_text(aes(label=Digit))+ ggtitle("MDS for MNIST using the SMC")
```




```
ggplot(mds2, aes(x=X1, y=X2, colour=Digit, label=Digit))+
  geom_text(aes(label=Digit))+ ggtitle("MDS for MNIST using the Jaccard index")
```



You can see that we get two different representations of the data that differ from each other, and from the PCA representation we computed in Chapter 4.

6.3 Non-metric MDS

In this chapter we have focused on classical MDS, which looks for points that have **Euclidean** distances close to the desired distances, where we measured close by minimizing the sum of squared errors (cf Equation (6.2)).

There is a much wider class of multidimensional scaling methods that do not use Euclidean distances, and do not judge the approximation quality using squared errors. These MDS methods can sometimes produce sets of coordinates that are more useful than classical MDS. However, it is only for classical MDS that there is a closed form solution. More general MDS methods rely upon numerical optimization to find a solution to the optimization problem. If you are interested, take a look at the R commands `isoMDS`, `Shepard`, and `sammon` from the `MASS` R package. Sammon mapping, for example, puts a greater emphasis on preserving smaller distances.

More generally still, there has been a lot of work on non-linear dimension reduction methods that try to find a low dimensional manifold upon which the data lie. Have a look here for some amazing visualizations and results using the MNIST dataset. These methods are not always easy to work with, and can require detailed user supervision in order to work well.

6.4 Exercises

1. In this question we will prove part 1. Theorem 6.1.

   i. Define
   $$b_{ij} = (\mathbf{x}_i - \bar{\mathbf{x}})^\top (\mathbf{x}_j - \bar{\mathbf{x}}), \quad i, j = 1, \ldots, n,$$
   and write $\mathbf{B} = (b_{ij})_{i,j=1}^n$. Prove that
   $$\mathbf{B} = (\mathbf{HX})(\mathbf{HX})^\top,$$
   where $\mathbf{H}$ is the $n \times n$ centering matrix and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ is the data matrix.

   ii. Show that $\mathbf{B}$ is positive semi-definite, i.e., explain why, for all $n \times 1$ vectors $\mathbf{a}$,
   $$\mathbf{a}^\top \mathbf{B} \mathbf{a} = \mathbf{a}^\top (\mathbf{HX})(\mathbf{HX})^\top \mathbf{a} \geq 0.$$

2. In this question we will prove part 2. of Theorem 6.1

   i. Prove Property 7. of Section 2.4. Specifically: if $\mathbf{A}$ is a symmetric $n \times n$ matrix, show that
   $$\mathbf{B} = \mathbf{HAH}$$
   has elements given by
   $$b_{ij} = a_{ij} - \bar{a}_{i+} - \bar{a}_{+j} + \bar{a}_{++}, \qquad i, j = 1, \ldots, n,$$
   where
   $$\bar{a}_{i+} = \frac{1}{n} \sum_{j=1}^n a_{ij}, \quad \bar{a}_{+j} = \frac{1}{n} \sum_{i=1}^n a_{ij}, \quad \text{and} \quad \bar{a}_{++} = \frac{1}{n^2} \sum_{i,j=1}^n a_{ij}.$$

   ii. Assume that $\mathbf{B}$ is positive semi-definite with k strictly positive eigenvalues and let
   $$\mathbf{B} = \sum_{j=1}^k \lambda_j \mathbf{u}_j \mathbf{u}_j^\top = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top,$$
   where $\mathbf{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_k\}$ and $\mathbf{U}$ is $n \times k$ and satisfies $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_k$. Now define a 'new' $n \times k$ data matrix
   $$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top = \mathbf{U} \mathbf{\Lambda}^{1/2}.$$
   Show that $b_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$ for all $i, j = 1, \ldots, n$.
   Hint: check that for $\mathbf{X}$ defined as above, $\mathbf{X}\mathbf{X}^\top = \mathbf{B}$.

iii. We now need to show that $\mathbf{D} = (d_{ij})$ represents the set of inter-point distances for this new data matrix. Recall that $a_{ij} = -d_{ij}^2/2$. Deduce that

$$(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j) = b_{ii} + b_{jj} - 2b_{ij};$$

and so, using the first part of this question, show that

$$(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j) = -2a_{ij} = d_{ij}^2.$$

Hence the new inter-point distances are the same as the original ones.

3. Suppose $\mathbf{X}$ is a $n \times p$ column centred data matrix. PCA looks at the spectral decomposition of the covariance matrix $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$, whereas MDS uses the spectral decomposition of the Gram matrix $\mathbf{X}\mathbf{X}^\top$. Using the SVD for $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, explain the relationship between these two methods.

4. Answer Q4 part (c) from the 2018-19 exam paper.

6.5 Computer Tasks

6.5.0.0.1 Task 1

The `eurodist` dataset in R gives the road distances between 21 European cities. Note that this is stored as a `dist` type of object, as outputted by the `dist` command, i.e. as a lower tri-diagonal matrix. `cmdscale` will take this directly as input.

```
data(eurodist)
eurodist
?eurodist
```

i. Perform multidimensional scaling on this data, and find a two-dimensional set of points which has interpoint distances approximately equal to the data.

ii. Plot these coordinates and label them with the city names. Does your plot look like the map of Europe?

iii. Is the distance matrix `eurodist` a Euclidean data matrix and how do you know? If it is not Euclidean, why do you think that might be?

iv. Create the Euclidean distance matrix from your set of 2-dimensional points. What is the Frobenius norm between this matrix and the original distance matrix? Use `cmdscale` to create a set of points in 3 dimensional space and recompute the distance matrix.

6.5.0.0.2 Task 2

Consider the synthetic data of 9 binary attributes on 11 cases.

```
df=structure(list(a = c(0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0), b = c(0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 1), c = c(0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 0), d = c(1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0), e = c(0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1), f = c(0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 1), g = c(0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0), h = c(1, 0, 0,
0, 0, 0, 0, 1, 1, 0, 0), i = c(0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
0)), class = "data.frame", row.names = c(NA, -11L), .Names = c("a",
"b", "c", "d", "e", "f", "g", "h", "i"))
df
```

```
##    a b c d e f g h i
## 1  0 0 0 1 0 0 0 1 0
## 2  0 0 0 0 0 0 1 0 0
## 3  0 0 0 0 1 1 1 0 0
## 4  0 0 0 0 0 0 1 0 0
## 5  1 0 1 0 0 0 0 0 0
## 6  0 0 0 0 0 0 0 0 1
## 7  0 1 0 1 0 0 0 0 0
## 8  1 0 1 0 0 0 0 1 0
## 9  0 1 0 0 0 0 0 1 1
## 10 0 0 1 1 0 0 0 0 1
## 11 0 1 0 0 1 1 0 0 0
```

i. Compute the Jaccard index and SMC similarity matrices for these data.

ii. Perform classical MDS for both similarity matrices, producing a plot of the coordinates (in 2d). Are the results similar?

6.5.0.0.3 Task 3

In this question we will look at data from 1888 on the fertility and socio-economic status of 47 French speaking provinces in Switzerland.

```
data(swiss)
head(swiss)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
## Moutier           85.8        36.5          12         7    33.77
## Neuveville        76.9        43.5          17        15     5.16
## Porrentruy        76.1        35.3           9         7    90.57
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
```

```
## Neuveville                20.6
## Porrentruy                26.6
```

```
?swiss
```

We will use MDS to find which provinces are similar to each other.

i. Compute the Euclidean distance matrix for these data.

ii. Use MDS to create a 2-dimensional representation of the data and plot these points, labelling them with the province name.

iii. Use MDS to create a 3-dimensional representation of the data. You can plot this using the `plot3d` command from the `rgl` package. See, for example, here.

```
library(rgl)
plot3d(mds3)
```

iii. MDS can be also used to reveal a hidden pattern in a correlation matrix. Find the correlation matrix, $\mathbf{R}$, of the swiss data. Perform MDS using $1 - \mathbf{R}$ as the distance matrix and plot the results. Positively correlated covariates are close together on the same side of the plot.

6.5.0.0.4 Task 4 (if you have time. . .)

Try MDS on the MNIST data but looking for a 3d representation. Colour the points by their digit label, and create some interactive 3d plots. Does this find useful structure in the data? And is it more informative than the 2d plots we created in the notes.

Read the description of more advance methods here. Pick one and find an R package that implements it and try it on the MNIST data.

Warning: The MNIST dataset is large, and so computations can take a long time if you use the full dataset. Thus I usually work with a selection of just 1000 images, which is enough to find interesting patterns in most cases.

Part III: Inference using the Multivariate Normal Distribution (MVN)

Part III of this module covers statistical inference based on the multivariate normal (MVN) distribution.

Chapter 7 focuses on classical distribution theory relating to the MVN distribution, including the Wishart distribution, which is defined on the set of symmetric positive definite matrices and is a natural generalisation of the χ^2 distribution. Another important distribution related to the MVN distribution is the Hotelling T^2 distribution, which is a multivariate analogue of the Student's t-distribution. The Wishart and Hotelling T^2 distributions then allow us to conduct hypothesis tests concerning vector means in 1-sample and 2-sample settings.

Chapter 10 is concerned with the multivariate linear model, in which the responses consist of random vectors rather than single random variables. Errors in this setting take the form of random vectors.

Chapter 7

The Multivariate Normal Distribution

The multivariate normal distribution (MVN) generalises the univariate normal distribution from scalar to vector random variables. It is important for a number of reasons:

1. It is entirely defined by its mean vector $\boldsymbol{\mu}$ and its covariance matrix $\boldsymbol{\Sigma}$.
2. Zero correlation implies independence.
3. Linear functions of multivariate normal vectors are also multivariate normal vectors.
4. The multivariate version of the Central Limit Theorem means that it appears naturally throughout statistics.
5. It has simple geometric properties, and is easy to work with mathematically.

In this chapter we'll define the MVN and look at some its properties. We'll then look at multivariate analogues of the t-test for comparing the mean of different populations. This will involve us defining two important the distributions:

- **Wishart** distribution which we can think of as the multivariate χ^2 distrbution, and which gives the distribution of sample covariance matrices.
- **Hotelling's** T^2 distribution, which is the multivariate version of Student's t-distribution.

The videos for this chapter, in the order I recommend you watch, are available at

- 7.1 Introduction to the multivariate normal distribution
- 7.1 Properties of the MVN distribution
- 7.1.4 Confidence ellipses for the MVN
- 7.4 Introduction to hypothesis testing
- 7.2 Wishart distribution
- 7.2.1 Properties of the Wishart distribution

- 7.3 Hotelling's T^2 distribution
- 7.4 Hypothesis tests with Hotelling's T^2 distribution

7.1 Definition and Properties of the MVN

7.1.1 Basics

Definition 7.1. A random vector $\mathbf{x} = (x_1, \ldots, x_p)^\top$ has a p-dimensional MVN distribution if and only if $\mathbf{a}^\top \mathbf{x}$ is a univariate normal random variable for all constant vectors $\mathbf{a} \in \mathbb{R}^p$.

In particular, note that the marginal distribution of each element of $\mathbf{x}$ has a uni-variate Gaussian distribution.

Notation: If $\mathbf{x} \in \mathbb{R}^p$ is MVN with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ then we write

$$\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Definition 7.2. If the population covariance matrix $\boldsymbol{\Sigma}$ $(p \times p)$ is positive definite (i.e. full rank), so that $\boldsymbol{\Sigma}^{-1}$ exists, then the **probability density function** (pdf) of the MVN distribution is given by

$$f(\mathbf{x}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

Here, I've used the notation $|\mathbf{A}| = \det(\mathbf{A})$.

If $p = 1$, so that $\mathbf{x} = x$, $\boldsymbol{\mu} = \mu$ and $\boldsymbol{\Sigma} = \sigma^2$, say, then the pdf simplifies to

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{|2\pi\sigma^2|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)(\sigma^2)^{-1}(x-\mu)\right) \\ &= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \end{aligned}$$

which is the familiar pdf of the univariate normal distribution $N(\mu, \sigma^2)$.

If $p > 1$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ then

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{((2\pi)^p \prod_{i=1}^p \sigma_i^2)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \\ &= \frac{1}{(2\pi)^{p/2} \prod_{i=1}^p \sigma_i} \exp\left(-\frac{1}{2}\sum_{i=1}^p \frac{(x_i-\mu_i)^2}{\sigma_i^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2\sigma_1^2}(x_1-\mu_1)^2\right)\right) \\ &\qquad \times \ldots \left(\frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{1}{2\sigma_p^2}(x_p-\mu_p)^2\right)\right) \end{aligned}$$

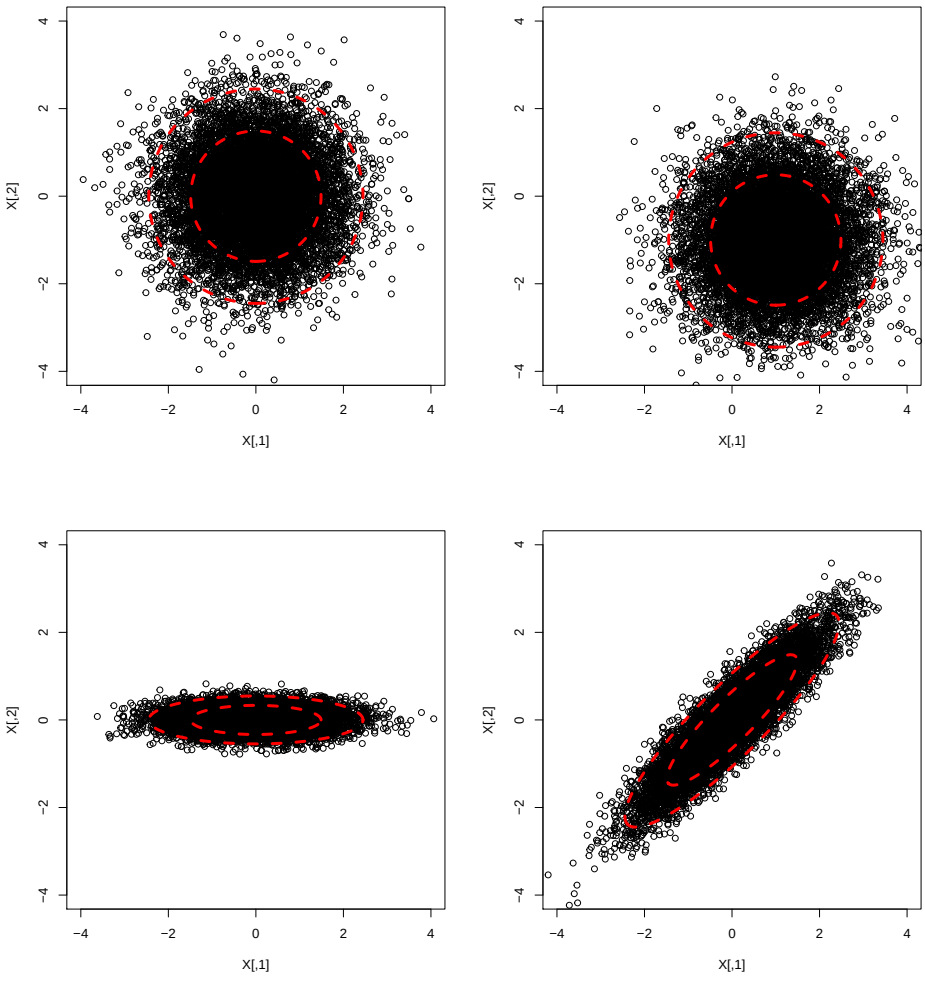
Thus, by the factorisation theorem for probability densities, the components of $\mathbf{x}$ have independent univariate normal distributions: $x_i \sim N(\mu_i, \sigma_i^2)$.

If $p = 2$ we can plot $f(\mathbf{x})$ using contour plots. Below, I've generated 1000 points from four different normal distributions using mean vectors

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_3 = \boldsymbol{\mu}_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

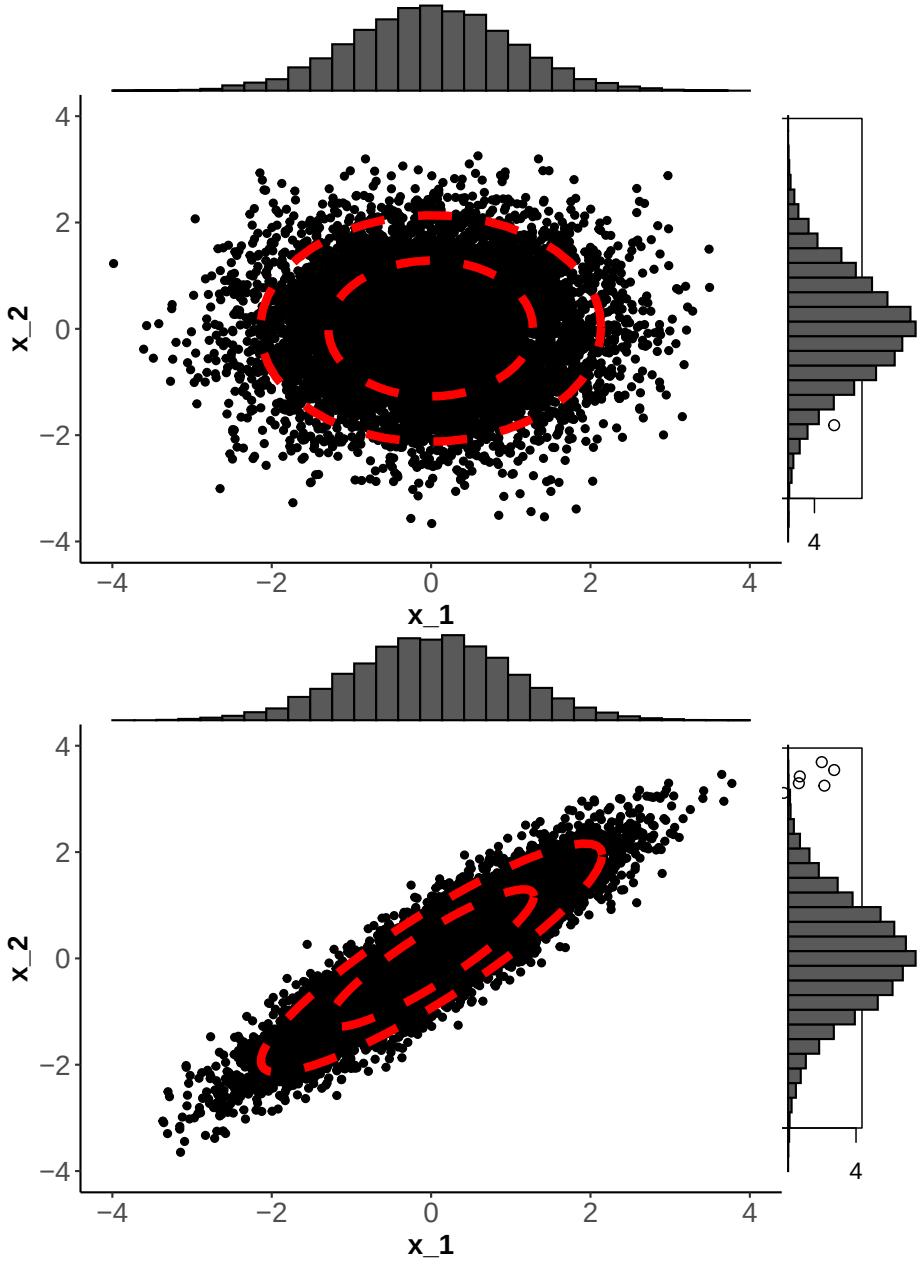
and covariance matrices

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{pmatrix} 1 & 0 \\ 0 & 0.05 \end{pmatrix}, \quad \boldsymbol{\Sigma}_4 = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}.$$



Note that the top left and bottom right plots have the **same marginal distribtions** for components x_1 and x_2, namely

$$x_1 \sim N(0, 1) \quad x_2 \sim N(0, 1)$$



The contours on each plot are obtained by finding values of $\mathbf{x}$ for which $f(\mathbf{x}) = c$. The constant c is chosen so that the the shapes (which we'll see below are ellipses) enclose 66% and 95% of the data.

7.1.2 Transformations

Proposition 7.1. *If* $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *then if*

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{c}, \text{ where } \mathbf{A} \in \mathbb{R}^{q \times p} \text{ and } \mathbf{c} \in \mathbb{R}^q \text{ are constant,}$$

then

$$\mathbf{y} \sim N_q(\mathbf{A}\boldsymbol{\mu} + \mathbf{c}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top).$$

Proof. Let $\mathbf{b} \in \mathbb{R}^q$ be a constant vector. Then

$$\mathbf{b}^\top \mathbf{y} = \mathbf{b}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{c} = \mathbf{a}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{c}$$

where $\mathbf{a}^\top = \mathbf{b}^\top \mathbf{A}$. Now $\mathbf{a}^\top \mathbf{x}$ is univariate normal for all $\mathbf{a}$ since $\mathbf{x}$ is MVN. Therefore $\mathbf{b}^\top \mathbf{y}$ is univariate normal for all $\mathbf{b}$, so $\mathbf{y}$ is MVN.

We can compute $\mathbb{E}(\mathbf{y}) = \mathbf{A}\boldsymbol{\mu} + \mathbf{c}$ and $\mathbb{V}\mathrm{ar}(\mathbf{y}) = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top$ using the properties listed in Section 1.3. □

This implies that a linear transformation of a MVN random variable is also MVN. We can use this result to prove two important corollaries. The first corollary is useful for simulating data from a general MVN distribution.

Corollary 7.1. *If* $\mathbf{x} \sim N_p(\mathbf{0}, \mathbf{I}_p)$ *and* $\mathbf{y} = \boldsymbol{\Sigma}^{1/2}\mathbf{x} + \boldsymbol{\mu}$ *then*

$$\mathbf{y} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Proof. Apply 7.1 with $\mathbf{A} = \boldsymbol{\Sigma}^{1/2}$ and $\mathbf{c} = \boldsymbol{\mu}$. Therefore

$$\mathbb{E}(\mathbf{y}) = \boldsymbol{\Sigma}^{1/2}\mathbf{0}_p + \boldsymbol{\mu} = \boldsymbol{\mu} \quad \text{and} \quad \mathbb{V}\mathrm{ar}(\mathbf{y}) = \boldsymbol{\Sigma}^{1/2}\mathbf{I}_p\boldsymbol{\Sigma}^{1/2} = \boldsymbol{\Sigma}.$$

□

The second corollary says that any MVN random variable can be transformed into standard form.

Corollary 7.2. *Suppose* $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, where* $\boldsymbol{\Sigma}$ *has full rank. Then*

$$\mathbf{y} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x} - \boldsymbol{\mu}) \sim N_p(\mathbf{0}, \mathbf{I}_p).$$

Proof. Apply Proposition 7.1 with $\mathbf{A} = \boldsymbol{\Sigma}^{-1/2}$ and $\mathbf{c} = -\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}$.

□

7.1.3 Independence

Proposition 7.2. *Two vectors* $\mathbf{x} \in \mathbb{R}^p$ *and* $\mathbf{y} \in \mathbb{R}^q$ *which are jointly multivariate normal are independent if and only if they are uncorrelated, i.e.* $\mathbb{C}\,\mathrm{ov}(\mathbf{x}, \mathbf{y}) = \mathbf{0}_{p,q}$.

Proof. The joint distribution of $\mathbf{x}$ and $\mathbf{y}$ can always be factorized as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x}).$$

If the conditional distribution for $\mathbf{y}$ given $\mathbf{x}$ does not depend upon $\mathbf{x}$, i.e., if $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y})$, then $\mathbf{y}$ and $\mathbf{x}$ are independent.

Suppose $\mathbf{x} \sim N_p(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{x}})$ and $\mathbf{y} \sim N_p(\boldsymbol{\mu}_\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{y},\mathbf{y}})$ are jointly normally distributed and that they are uncorrelated. Thus we can write

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim N_{p+q}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$$

where

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_\mathbf{x} \\ \boldsymbol{\mu}_\mathbf{y} \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{x}} & \mathbf{0}_{p,q} \\ \mathbf{0}_{q,p} & \boldsymbol{\Sigma}_{\mathbf{z},\mathbf{z}} \end{pmatrix}.$$

Now

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})} \\ &\propto \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^\top \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{x}}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) - \frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})^\top \boldsymbol{\Sigma}_{\mathbf{y},\mathbf{y}}(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})\right)}{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^\top \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{x}}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})\right)} \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})^\top \boldsymbol{\Sigma}_{\mathbf{y},\mathbf{y}}(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y})\right) \\ &\propto p(\mathbf{y}) \end{aligned}$$

So $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y})$, i.e. $p(\mathbf{y}|\mathbf{x})$ is not a function of $\mathbf{x}$, and thus $\mathbf{x}$ and $\mathbf{y}$ are independent. □

Proposition 7.2 means that zero correlation implies independence for the MVN distribution. This is not true in general for other distributions.

Note: It is important that $\mathbf{x}$ and $\mathbf{y}$ are **jointly** multivariate normal. For example, suppose $x \sim N(0, 1)$. Let

$$z = \begin{cases} 1 \text{ with probability } \frac{1}{2} \\ -1 \text{ otherwise} \end{cases}$$

and let $y = zx$. Then clearly y is also a normal random variable: $y \sim N(0, 1)$. In addition, note that

$$\mathbb{C}\,\mathrm{ov}(x, y) = \mathbb{E}(xy) = \mathbb{E}(x^2)\mathbb{E}(z) = 0$$

so that x and y are uncorrelated.

However, x and y are clearly not independent: if you tell me x, then I know $y = x$ or $y = -x$.

7.1.4 Confidence ellipses

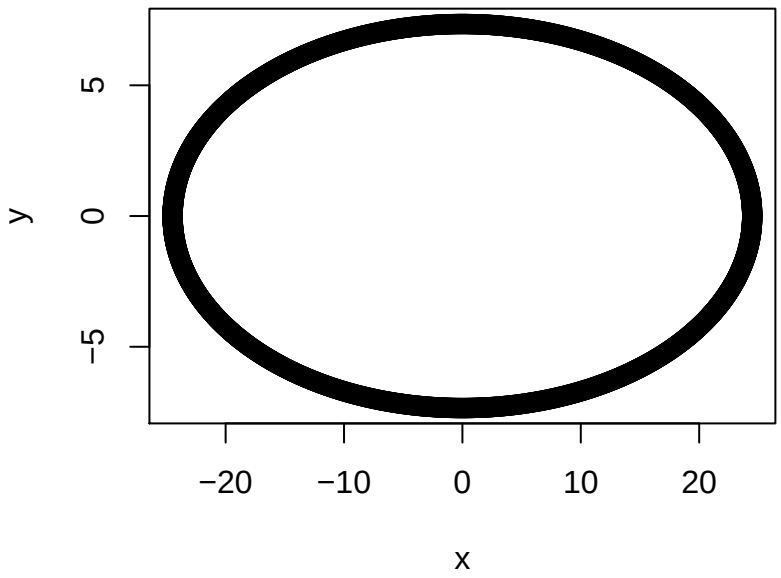
The contours in the plots of the bivariate normal samples shown above are ellipses. They were defined to lines of consant density, i.e., by $f(\mathbf{x}) = c$, which implies

$$(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = c' \tag{7.1}$$

for some constant c'. To see that this is the equation of an ellipse, note that a standard ellipse in $\mathbb{R}^2$ is given by the equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (a, b > 0) \tag{7.2}$$

and recall that a standard ellipse has axes of symmetry given by the x-axis and y-axis (if $a > b$, the x-axis is the major axis, and the y-axis the minor axis). For example, $a = 10, b = 3$ gives the ellipse:



If we define $\mathbf{A} = \begin{pmatrix} a^2 & 0 \\ 0 & b^2 \end{pmatrix}$ and write $\mathbf{x} = \binom{x}{y}$, then Equation (7.2) can be written in the form

$$\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x} = c'.$$

To shift the centre of the ellipse from the origin to the point $\boldsymbol{\mu}$ we modify the equation to be

$$(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{A}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = c'.$$

What if instead of using a diagonal matrix $\mathbf{A}$, we use a non-diagonal matrix $\mathbf{\Sigma}$ as in Equation (7.1)? If $\mathbf{\Sigma}$ has spectral decomposition $\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, then

$$\begin{aligned}(\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) &= (\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^\top(\mathbf{x}-\boldsymbol{\mu}) \\ &= \mathbf{y}^\top\mathbf{\Lambda}^{-1}\mathbf{y} \\ &= \frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2}\end{aligned}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2)$ is a diagonal matrix of eigenvalues, and $\mathbf{y} = \mathbf{V}^\top(\mathbf{x}-\mathbf{u})$. Because $\mathbf{V}$ is an orthogonal matrix (a rotation), we can see that this is the equation of a standard ellipse when using the eigenvectors as the coordinate system. Or in other words, it is an ellipse with major axis given by the first eigenvector, and minor axis given by the second eigenvector, centered around $\boldsymbol{\mu}$.

Analogous results for ellipsoids and quadratic forms hold in three and higher dimensions.

The term $(\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ will be important later in the chapter. The following proposition gives its distribution:

Proposition 7.3. *If $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \mathbf{\Sigma})$ and $\mathbf{\Sigma}$ is positive definite then*

$$(\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \sim \chi^2_p.$$

Proof. Define $\mathbf{y} = \mathbf{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu})$ so

$$\begin{aligned}(\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) &= \left(\mathbf{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu})\right)^\top\left(\mathbf{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu})\right) \\ &= \mathbf{y}^\top\mathbf{y} = \sum_{i=1}^p y_i^2\end{aligned}$$

By Corollary 7.2, $\mathbf{y} \sim N_p(\mathbf{0}, \mathbf{I}_p)$, and so the components of $\mathbf{y}$ have independent univariate normal distributions with mean 0 and variance 1. Recall from univariate statistics that if $z \sim N(0,1)$ then $z^2 \sim \chi^2_1$ and if $z_1, \ldots, z_n$ are iid $N(0,1)$ then $\sum_{i=1}^n z_i^2 \sim \chi^2_n$. It therefore follows that

$$\sum_{i=1}^p y_i^2 \sim \chi^2_p.$$

□

Proposition 7.3 means we can calculate the probability

$$\mathbb{P}\left((\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) < k\right),$$

which is the probability of $\mathbf{x}$ lying within a particular ellipsoid. We often use this to draw confidence ellipses, which are ellipses that we expect to contain some specified proportion of the random samples (95% say).

7.1.5 Sampling results for the MVN

In this section we present two important results which are natural generalisations of what happens in the univariate case.

Proposition 7.4. *If $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is an IID random sample from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the sample mean and sample variance matrix*

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i, \quad \mathbf{S} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

are independent.

Proof. From Proposition 7.1 we can see that if $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then $\bar{\mathbf{x}} \sim N_p(\boldsymbol{\mu}, n^{-1}\boldsymbol{\Sigma})$. Let $\mathbf{y}_i = \mathbf{x}_i - \bar{\mathbf{x}}$. Then

$$\begin{aligned}
\mathbb{C}\,\mathrm{ov}(\bar{\mathbf{x}}, \mathbf{y}_i) &= \mathbb{C}\,\mathrm{ov}(\bar{\mathbf{x}}, \mathbf{x}_i - \bar{\mathbf{x}}) \\
&= \mathbb{C}\,\mathrm{ov}(\bar{\mathbf{x}}, \mathbf{x}_i) - \mathbb{C}\,\mathrm{ov}(\bar{\mathbf{x}}, \bar{\mathbf{x}}) \\
&= n^{-1}\sum_{j=1}^{n} \left\{\mathbb{E}[(\mathbf{x}_j - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top]\right\} \\
&\qquad - \mathbb{E}[(\bar{\mathbf{x}} - \boldsymbol{\mu})(\bar{\mathbf{x}} - \boldsymbol{\mu})^\top] \\
&= n^{-1}\boldsymbol{\Sigma} - n^{-1}\boldsymbol{\Sigma} \\
&= \mathbf{0}_{p,p}.
\end{aligned}$$

Thus Proposition 7.2 gives that $\bar{\mathbf{x}}$ and $\mathbf{y}_i$ are independent, and therefore $\bar{\mathbf{x}}$ and

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^\top = n^{-1}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$$

are independent.

□

Recall from above that if $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is a random sample from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then

$$\bar{\mathbf{x}} \sim N_p(\boldsymbol{\mu}, \frac{1}{n}\boldsymbol{\Sigma}).$$

This result is also approximately true for large samples from non-normal distributions, as is now stated in the multivariate central limit theorem.

Proposition 7.5. ***Central limit theorem*** *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbb{R}^p$ be a sample of independent and identically distributed random vectors from a distribution with mean $\boldsymbol{\mu}$ and finite variance matrix $\boldsymbol{\Sigma}$. Then asymptotically as $n \to \infty$, $\sqrt{n}(\bar{\mathbf{x}} - \boldsymbol{\mu})$ converges in distribution to $N_p(\mathbf{0}_p, \boldsymbol{\Sigma})$.*

Proof. Beyond the scope of this module. □

7.2 The Wishart distribution

In univariate statistics the χ^2 distribution plays an important role in inference related to the univariate normal, e.g. in the definition of Student's t-distribution. The **Wishart distribution** is a multivariate generalisation of the univariate χ^2 distribution, and it plays an analogous role in multivariate statistics.

In this section we introduce the Wishart distribution and show that for MVN random variables, the sample covariance matrix $\mathbf{S}$ has a Wishart distribution.

Definition 7.3. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be an IID random sample from $N_p(\mathbf{0}, \mathbf{\Sigma})$. Then

$$\mathbf{M} = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top \in \mathbb{R}^{p \times p}$$

is said to have a Wishart distribution with n degrees of freedom and scale matrix $\mathbf{\Sigma}$. We write this as

$$\mathbf{M} \sim W_p(\mathbf{\Sigma}, n)$$

and refer to $W_p(\mathbf{I}_p, n)$ as a standard Wishart distribution.

Note:

- $W_p(\mathbf{\Sigma}, n)$ is a probability distribution on the set of $p \times p$ symmetric non-negative definite random matrices.
- Recall that if $z_1, \ldots, z_n \sim N(0, 1)$, then

  $$\sum_{i=1}^{n} z_i^2 \sim \chi^2_n.$$

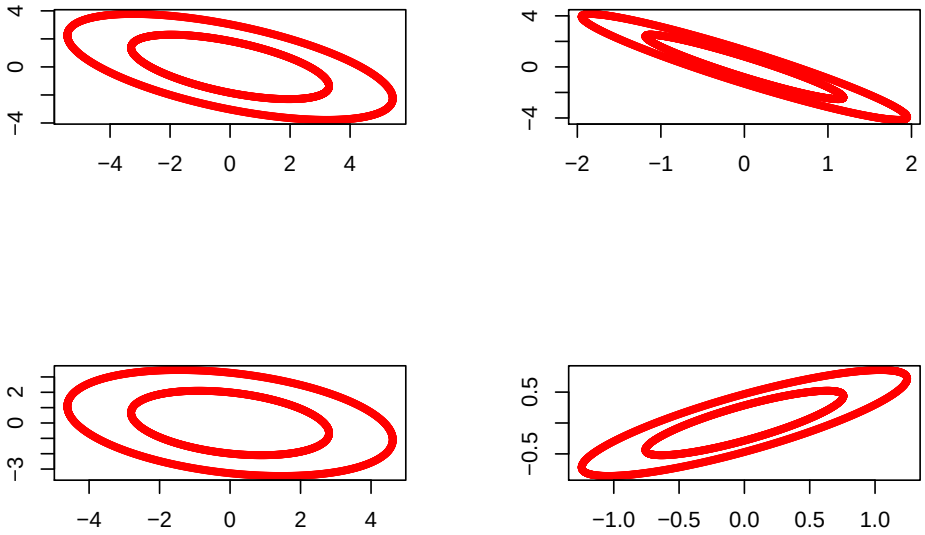
  Thus we can see that the Wishart distribution arises from the same kind of process: it is the sum of zero mean (multivariate) normal random variables squared.
- In particular, note that when $p = 1$, $W_1(1, n)$ is the χ^2_n distribution and $W_1(\sigma^2, n)$ is the $\sigma^2 \chi^2_n$ distribution.
- If $\mathbf{X}$ is the usual $n \times p$ matrix with rows $\mathbf{x}_i^\top$, then

  $$\mathbf{M} = \mathbf{X}^\top \mathbf{X}.$$

We can sample from the Wishart distribution in R using the `rWishart` command. For example, setting $\mathbf{\Sigma} = \mathbf{I}_2$ and using 2 degrees of freedom, we can generate 4 random samples $\mathbf{M}_1, \ldots, \mathbf{M}_4 \sim W_2(\mathbf{I}_2, 2)$ as follows:

```
out <- rWishart(n=4, df=2, Sigma=diag(1,2))
```

Visualizing these by plotting the ellipses with $\mathbf{x}^\top \mathbf{M}_i \mathbf{x} = c$ for some constant c, we can see the variability in these random matrices:



Proposition 7.6. *Let* $\mathbf{M} \sim W_p(\mathbf{\Sigma}, n)$. *Then*

$$\mathbb{E}\mathbf{M} = n\mathbf{\Sigma}$$

and if the ij^{th} *element of* $\mathbf{\Sigma}$ *is* σ_{ij}, *and the* ij^{th} *element of* $\mathbf{M}$ *is* m_{ij}, *then*

$$\mathbb{V}\mathrm{ar}(m_{ij}) = n\left(\sigma_{ij}^2 + \sigma_{ii}\sigma_{jj}\right)$$

7.2.1 Properties

We now use the definition of $W_p(\mathbf{\Sigma}, n)$ to prove some important results.

Proposition 7.7. *If* $\mathbf{M} \sim W_p(\mathbf{\Sigma}, n)$ *and* $\mathbf{A}$ *is a fixed* $q \times p$ *matrix, then*

$$\mathbf{AMA}^\top \sim W_q\left(\mathbf{A\Sigma A}^\top, n\right).$$

Proof. From the definition, let $\mathbf{M} = \sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$, where $\mathbf{x}_i \sim N_p(\mathbf{0}, \mathbf{\Sigma})$. Then

$$\begin{aligned}\mathbf{AMA}^\top &= \mathbf{A}\left(\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{A}^\top \\ &= \sum_{i=1}^n (\mathbf{Ax}_i)(\mathbf{Ax}_i)^\top = \sum_{i=1}^n \mathbf{y}_i\mathbf{y}_i^\top\end{aligned}$$

where $\mathbf{y}_i = \mathbf{Ax}_i \sim N_q(\mathbf{0}, \mathbf{A\Sigma A}^\top)$, by Proposition 7.1. Now we apply the definition of the Wishart distribution to $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and, hence, $\sum_{i=1}^n \mathbf{y}_i\mathbf{y}_i^\top \sim W_q\left(\mathbf{A\Sigma A}^\top, n\right)$. □

Proposition 7.8. *If* $\mathbf{M} \sim W_p(\mathbf{\Sigma}, n)$ *and* $\mathbf{a}$ *is a fixed* $p \times 1$ *vector then*

$$\mathbf{a}^\top\mathbf{Ma} \sim \left(\mathbf{a}^\top\mathbf{\Sigma a}\right)\chi_n^2.$$

Note that an alternative way to write this is as

$$\frac{\mathbf{a}^\top \mathbf{M}\mathbf{a}}{\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a}} \sim \chi^2_n.$$

Proof. Applying Proposition 7.7 with $\mathbf{A} = \mathbf{a}^\top$, we see $\mathbf{a}^\top \mathbf{M}\mathbf{a} \sim W_1(\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a}, n)$. If we let $z_i \sim N(0,1)$, and $\sigma = (\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a})^{\frac{1}{2}}$, then $\sigma z_i \sim N(0, \mathbf{a}^\top \mathbf{\Sigma}\mathbf{a})$. Thus

$$\begin{aligned}\sum_{i=1}^n \sigma^2 z_i^2 &\sim W_1(\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a}, n) \quad \text{by the definition of the Wishart distribution} \\ &= \sigma^2 \sum_{i=1}^n z_i \\ &\sim (\mathbf{a}^\top \mathbf{\Sigma}\mathbf{a})\chi^2_n \quad \text{by the definition of} \chi^2.\end{aligned}$$

□

Proposition 7.9. *If* $\mathbf{M}_1 \sim W_p(\mathbf{\Sigma}, n_1)$ *and* $\mathbf{M}_2 \sim W_p(\mathbf{\Sigma}, n_2)$ *are independent then*

$$\mathbf{M}_1 + \mathbf{M}_2 \sim W_p(\mathbf{\Sigma}, n_1 + n_2).$$

Proof. From the definition, let $\mathbf{M}_1 = \sum_{i=1}^{n_1} \mathbf{x}_i \mathbf{x}_i^\top$ and let $\mathbf{M}_2 = \sum_{i=n_1+1}^{n_1+n_2} \mathbf{x}_i \mathbf{x}_i^\top$, where $\mathbf{x}_i \sim N_p(\mathbf{0}, \mathbf{\Sigma})$, then $\mathbf{M}_1 + \mathbf{M}_2 = \sum_{i=1}^{n_1+n_2} \mathbf{x}_i \mathbf{x}_i^\top \sim W_p(\mathbf{\Sigma}, n_1 + n_2)$ by the definition of the Wishart distribution. □

7.2.2 Cochran's theorem

Our next result is known as Cochran's theorem. We use Cochran's theorem to show that sample covariance matrices have a scaled Wishart distribution.

First though, recall the definition of projection matrices from Section 2.3.3. Namely, that $\mathbf{P}$ is a projection matrix if $\mathbf{P}^2 = \mathbf{P}$.

Theorem 7.1. ***(Cochran's Theorem)*** *Suppose* $\overset{n\times n}{\mathbf{P}}$ *is a projection matrix of rank* r*. Assume that* $\mathbf{X}$ *is an* $n \times p$ *data matrix with IID rows that have a common* $N_p(\mathbf{0}_p, \mathbf{\Sigma})$ *distribution, where* Σ *has full rank* p*. Note the identity*

$$\mathbf{X}^\top \mathbf{X} = \mathbf{X}^\top \mathbf{P}\mathbf{X} + \mathbf{X}^\top (\mathbf{I}_n - \mathbf{P})\mathbf{X}. \tag{7.3}$$

Then

$$\mathbf{X}^\top \mathbf{P}\mathbf{X} \sim W_p(\mathbf{\Sigma}, r), \qquad \mathbf{X}^\top (\mathbf{I}_n - \mathbf{P})\mathbf{X} \sim W_p(\mathbf{\Sigma}, n - r), \tag{7.4}$$

and $\mathbf{X}^\top \mathbf{P}\mathbf{X}$ *and* $\mathbf{X}^\top (\mathbf{I}_n - \mathbf{P})\mathbf{X}$ *are independent.*

We'll prove this result below. Let's first understand why it is useful.

Proposition 7.10. *If $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is an IID sample from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then*

$$n\mathbf{S} = \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \sim W_p(\boldsymbol{\Sigma}, n-1).$$

Proof. Let $\mathbf{P} = \mathbf{H} \equiv \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$, the $n \times n$ centering matrix, where $\mathbf{1}_n$ is the $n \times 1$ vector of ones.

$\mathbf{H}$ is a projection matrix (property 1. of 2.4), and clearly, $\mathbf{I}_n - \mathbf{P} = n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$ has rank 1, and thus $\mathbf{H}$ must have rank $n-1$. Therefore, using Cochran's Theorem (7.1),

$$\mathbf{X}^\top \mathbf{H}\mathbf{X} \sim W_p(\boldsymbol{\Sigma}, n-1).$$

But

$$\mathbf{X}^\top \mathbf{H}\mathbf{X} = n\mathbf{S},$$

(Property 6. in Section 2.4) and consequently, $n\mathbf{S} \sim W_p(\boldsymbol{\Sigma}, n-1)$, as required. □

Thus, sample covariance matrices have a scaled Wishart distribution. This result will be key in the next section, as it will allow us to compute the sampling distribution of a test statistic that we will then use in hypothesis test.

We will now prove Cochran's theorem.

Proof. **Non-examinable**

We first prove the result for the case $\boldsymbol{\Sigma} = \mathbf{I}_p$.

Using the Spectral Decomposition Theorem 3.3 and noting that the eigenvalues of projection matrices must be either 0 or 1, we can write

$$\mathbf{P} = \sum_{j=1}^{r} \mathbf{v}_j\mathbf{v}_j^\top \qquad \text{and} \qquad (\mathbf{I}_n - \mathbf{P}) = \sum_{j=r+1}^{n} \mathbf{v}_j\mathbf{v}_j^\top$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ are mutually orthogonal unit vectors. Then

$$\begin{aligned}\mathbf{X}^\top \mathbf{P}\mathbf{X} &= \mathbf{X}^\top \left(\sum_{j=1}^{r} \mathbf{v}_j\mathbf{v}_j^\top \right) \mathbf{X} \\ &= \sum_{j=1}^{r} \mathbf{X}^\top \mathbf{v}_j\mathbf{v}_j^\top \mathbf{X} = \sum_{j=1}^{r} \mathbf{y}_j\mathbf{y}_j^\top, \end{aligned} \tag{7.5}$$

and similarly,

$$\mathbf{X}^\top (\mathbf{I}_n - \mathbf{P})\mathbf{X} = \sum_{j=r+1}^{n} \mathbf{y}_j\mathbf{y}_j^\top, \tag{7.6}$$

where $\mathbf{y}_j = \mathbf{X}^\top \mathbf{v}_j$ is a $p \times 1$ vector.

Claim The $\mathbf{y}_j$ are iid multivariate normal random variables:

$$\mathbf{y}_j \sim N_p(\mathbf{0}_p, \mathbf{I}_p).$$

If the claim is true, then it immediately follows from the definition of the Wishart distribution that (7.5) has a Wishart $W_p(\mathbf{I}_p, r)$ distribution and (7.6) has a Wishart $W_p(\mathbf{I}_p, n-r)$ distribution. Moreover they are independent becasue the $\mathbf{y}_j$ are all independent.

Then to prove the general case with covariance matrix $\mathbf{\Sigma}$, note that if $\mathbf{x}_i \sim N_p(\mathbf{0}, \mathbf{\Sigma})$, then we can write $\mathbf{x}_i = \mathbf{\Sigma}^{1/2}\mathbf{z}_i$ where $\mathbf{z}_i \sim N_p(\mathbf{0}, \mathbf{I}_p)$.

Thus

$$\begin{aligned}\mathbf{X}^\top \mathbf{P}\mathbf{X} &= \mathbf{\Sigma}^{1/2}\mathbf{Z}^\top \mathbf{P}\mathbf{Z}\mathbf{\Sigma}^{1/2} \\ &\sim \mathbf{\Sigma}^{1/2} W_p(\mathbf{I}_p, r)\mathbf{\Sigma}^{1/2} \text{ by the result above} \\ &\sim W_p(\mathbf{\Sigma}, r)\end{aligned}$$

where the final line follows by Proposition 7.7. Here, $\mathbf{X}$ and $\mathbf{Z}$ are matrices with rows given by $\mathbf{x}_i$ and $\mathbf{z}_i$ respectively.

To complete the proof it only remains to prove the claim that $\mathbf{y}_j \sim N_p(\mathbf{0}_p, \mathbf{I}_p)$.

We can immediately see that the $\mathbf{y}_j$ must be MVN of dimension p, and that they have mean vector $\mathbf{0}_p$. To see the covariance and independence parts, note that the k^{th} element of $\mathbf{y}_j$ is

$$y_{jk} = \sum_{i=1}^{n} x_{ik} v_{ji}$$

and so the k, l^{th} element of the covariance matrix between $\mathbf{y}_j$ and $\mathbf{y}_{j'}$ is

$$\begin{aligned}\mathbb{E}(y_{jk} y_{j'l}) &= \mathbb{E}(\sum_{i=1}^{n} x_{ik} v_{ji} \sum_{i'=1}^{n} x_{i'l} v_{j'i'}) \\ &= \sum_{i=1}^{n}\sum_{i'=1}^{n} v_{ji}\mathbb{E}(x_{ik}x_{i'l})v_{j'i'} \\ &= \begin{cases} 0 & \text{if } k \neq l \text{ as } x_{ik} \text{ independent of } x_{il} \\ \sum_{i=1}^{n} v_{ji}v_{j'i} & \text{if } k = l \text{ as } x_{ik} \text{ is independent of } x_{i'k} \text{ for } i \neq i'. \end{cases}\end{aligned}$$

Finally

$$\sum_{i=1}^{n} v_{ji}v_{j'i} = \mathbf{v}_j^\top \mathbf{v}_{j'}$$
$$= \begin{cases} 1 & \text{if } j = j' \\ 0 & \text{otherwise.} \end{cases}$$

Thus $\mathbb{C}\,\mathrm{ov}(\mathbf{y}_j, \mathbf{y}_{j'}) = \mathbf{0}_{p\times p}$ for $j \neq j'$ and $\mathbb{V}\,\mathrm{ar}(\mathbf{y}_j) = \mathbf{I}_p$. Thus we have proved the claim once we recall that uncorrelated implies independence for multivariate normal random variables. □

7.3 Hotelling's T^2 distribution

Recall that in univariate statistics, Student's t-distribution appears as the sampling distribution of $\frac{\bar{x}-\mu}{s/\sqrt{n}}$, which is used for hypothesis tests and constructing confidence intervals.

Hotelling's T^2 distribution is the multivariate analogue of Student's t-distribution. It plays an important role in multivariate hypothesis testing and confidence region construction, just as the Student t-distribution does in the univariate setting.

Definition 7.4. Suppose $\mathbf{x} \sim N_p(\mathbf{0}, \mathbf{I}_p)$ and $\mathbf{M} \sim W_p(\mathbf{I}_p, n)$ are independent, then the quantity

$$\tau^2 = n\mathbf{x}^\top \mathbf{M}^{-1}\mathbf{x}$$

is said to have **Hotelling's T^2 distribution** with parameters p and n. We write this as

$$\tau^2 \sim T^2(p, n).$$

This is reminiscent of the definition of the Student t-distribution: if $x \sim N(0,1)$ and $v \sim \chi^2_n$, then

$$T = \frac{x}{\sqrt{v/n}} \sim t_n.$$

Hotelling's T^2 distribution looks similar (albeit working with the square):a MVN random variable 'divided' by a Wishart r.v. divided by the degrees of freedom.

We can generalise the definition with the following result.

Proposition 7.11. *Suppose $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{M} \sim W_p(\boldsymbol{\Sigma}, n)$ are independent and $\boldsymbol{\Sigma}$ has full rank p. Then*

$$n(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{M}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \sim T^2(p, n).$$

Proof. Define $\mathbf{y} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu})$. Then, by Corollary 7.2, $\mathbf{y} \sim N_p(\mathbf{0}, \mathbf{I}_p)$. Further, let $\mathbf{Z} = \boldsymbol{\Sigma}^{-1/2}\mathbf{M}\boldsymbol{\Sigma}^{-1/2}$ then $\mathbf{Z} \sim W_p(\mathbf{I}_p, n)$ by applying 7.7 with $\mathbf{A} = \boldsymbol{\Sigma}^{-1/2}$. From the definition, $n\mathbf{y}^\top\mathbf{Z}^{-1}\mathbf{y} \sim T^2(p,n)$ and

$$\begin{aligned} n\mathbf{y}^\top\mathbf{Z}^{-1}\mathbf{y} &= n(\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{1/2}\mathbf{M}^{-1}\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\mu}) \\ &= n(\mathbf{x}-\boldsymbol{\mu})^\top\mathbf{M}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \end{aligned}$$

so the result is proved. □

This result gives rise to an important corollary used in hypothesis testing when $\boldsymbol{\Sigma}$ is unknown.

Corollary 7.3. *If $\bar{\mathbf{x}}$ and $\mathbf{S}$ are the mean and covariance matrix based on a sample of size n from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then*

$$(n-1)(\bar{\mathbf{x}}-\boldsymbol{\mu})^\top\mathbf{S}^{-1}(\bar{\mathbf{x}}-\boldsymbol{\mu}) \sim T^2(p, n-1).$$

Proof. We have seen earlier that $\bar{\mathbf{x}} \sim N_p(\boldsymbol{\mu}, \frac{1}{n}\boldsymbol{\Sigma})$. Let $\mathbf{x}^* = n^{1/2}\bar{\mathbf{x}}$ and let $\boldsymbol{\mu}^* = n^{1/2}\boldsymbol{\mu}$. Then $\mathbf{x}^* = n^{1/2}\bar{\mathbf{x}} \sim N_p(\boldsymbol{\mu}^*, \boldsymbol{\Sigma})$.

From Proposition 7.10 we know $n\mathbf{S} \sim W_p(\boldsymbol{\Sigma}, n-1)$, and from Theorem 7.4 we know $\bar{\mathbf{x}}$ and $\mathbf{S}$ are independent. Applying Proposition 7.11 with $\mathbf{x} = \mathbf{x}^*$ and $\mathbf{M} = n\mathbf{S}$ we obtain

$$(n-1)(\mathbf{x}^*-\boldsymbol{\mu}^*)^\top(n\mathbf{S})^{-1}(\mathbf{x}^*-\boldsymbol{\mu}^*) \sim T^2(p, n-1),$$

and given $\mathbf{x}^*-\boldsymbol{\mu}^* = n^{1/2}(\mathbf{x}-\boldsymbol{\mu})$ then

$$\begin{aligned} &(n-1)(\mathbf{x}^*-\boldsymbol{\mu}^*)^\top(n\mathbf{S})^{-1}(\mathbf{x}^*-\boldsymbol{\mu}^*) \\ &\quad= (n-1)n^{1/2}(\bar{\mathbf{x}}-\boldsymbol{\mu})^\top n^{-1}\mathbf{S}^{-1}n^{1/2}(\bar{\mathbf{x}}-\boldsymbol{\mu}) \\ &\quad= (n-1)(\bar{\mathbf{x}}-\boldsymbol{\mu})^\top\mathbf{S}^{-1}(\bar{\mathbf{x}}-\boldsymbol{\mu}). \end{aligned}$$

□

Hotelling's T^2 distribution is not often included in statistical tables but the next result tells us that Hotelling's T^2 is a scale transformation of an F distribution.

Proposition 7.12. *If $\tau^2 \sim T^2(p, n-1)$ then*

$$\gamma^2 = \frac{n-p}{(n-1)p}\tau^2 \sim F_{p,n-p}.$$

Proof. Beyond the scope of the module. □

We can apply this result to the previous corollary.

Corollary 7.4. *If $\tau^2 = (n-1)(\bar{\mathbf{x}}-\boldsymbol{\mu})^\top\mathbf{S}^{-1}(\bar{\mathbf{x}}-\boldsymbol{\mu})$ then*

$$\gamma^2 = \frac{n-p}{p}(\bar{\mathbf{x}}-\boldsymbol{\mu})^\top\mathbf{S}^{-1}(\bar{\mathbf{x}}-\boldsymbol{\mu}) \sim F_{p,n-p}.$$

We'll use this result to do hypothesis tests.

7.4 Inference based on the MVN

In univariate statistical analysis, you will have seen how to do hypothesis testing for the mean of a population.

1. In the case where you have a single sample $x_1, \ldots, x_n$ which come from a population with known variance σ^2 we use a z-test when testing hypotheses such as $H_0 : \mu = \mu_1$.
2. When the variance of the population, σ^2, is unknown, then we have to use a t-test.
3. When we have two samples, $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$, we use either a paired or an unpaired t-test.

We now develop analogous results in the multivariate case. The role of the Student t-distribution will be played by Hotelling's T^2, and the role of the χ^2 is played by the Wishart distribution.

The next three subsections deal with the multivariate equivalent of the three situations listed above. Before we do, lets quickly recap how hypothesis testing works:

Recap of hypothesis testing framework

Suppose that we have a null hypothesis H_0 represented by a completely specified model and that we wish to test this hypothesis using data $x_1, \ldots, x_n$. We proceed as follows

1. Assume H_0 is true.
2. Find a test statistic $T(x_1, \ldots, x_n)$ for which large values indicate departure from H_0.
3. Calculate the theoretical sampling distribution of T under H_0.
4. The observed value $T_{obs} = T(x_1, \ldots, x_n)$ of the test statistic is compared with the distribution of T under H_0. Then either
   - (Neyman-Pearson) reject H_0 if $T_{obs} > c$. Here c is chosen so that $\mathbb{P}(T \geq c|H_0) = \alpha$ where α is the **size** of the test, i.e., $\mathbb{P}(\text{reject } H_0|H_0 \text{ true}) = \alpha$.
   - (Fisherian) compute the p-value $p = \mathbb{P}(T \geq T_{obs}|H_0)$ and report it. This represents the strength of evidence against H_0.

7.4.1 Σ known

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a random sample from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^\top$. Suppose we wish to conduct the following hypothesis test

$$H_0 : \boldsymbol{\mu} = \mathbf{a} \text{ vs } H_1 : \boldsymbol{\mu} \neq \mathbf{a}$$

where $\mathbf{a}$ is fixed and pre-specified. Let's first **assume that $\boldsymbol{\Sigma}$ is known**. This will result in the multivariate analogue of the z-test.

One approach would be to conduct p separate univariate z-tests tests with null hypotheses

$$H_0 : \mu_i = a_i \qquad \text{vs.} \qquad H_1 : \mu_i \neq a_i, \quad \text{for } i = 1, \ldots, p.$$

However, this ignores possible correlations between the variables - see the example for a situation in which this can make a difference.

A better approach is to conduct a single (multivariate) hypothesis test using the test statistic

$$\zeta^2 = n(\bar{\mathbf{x}} - \mathbf{a})^\top \boldsymbol{\Sigma}^{-1}(\bar{\mathbf{x}} - \mathbf{a}).$$

We need to compute the distribution of ζ^2 when H_0 is true. Note that

$$\zeta^2 = (n^{1/2}\bar{\mathbf{x}} - n^{1/2}\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(n^{1/2}\bar{\mathbf{x}} - n^{1/2}\boldsymbol{\mu}).$$

Recall that $\bar{\mathbf{x}} \sim N_p(\boldsymbol{\mu}, \frac{1}{n}\boldsymbol{\Sigma})$, and that therefore $n^{1/2}\bar{\mathbf{x}} \sim N_p(n^{1/2}\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Applying Proposition
Proposition 7.3 we thus see that

$$\zeta^2 \sim \chi^2_p$$

when H_0 is true.

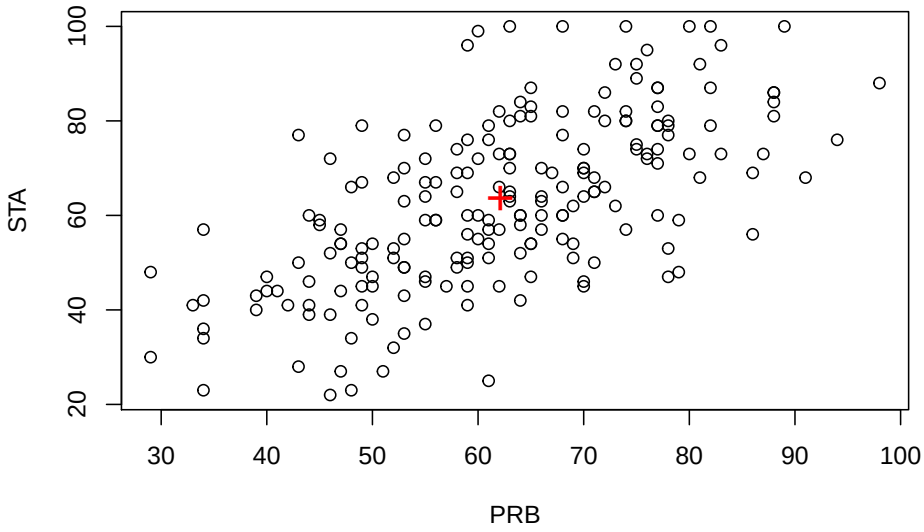
Thus to conduct the hypothesis test, we compute ζ^2 for the observed data, and if the value is large compared to a χ^2_p distribution, we reject H_0.

- The Neyman-Pearson approach is to define a critical region and reject H_0 if $\zeta^2 > \chi^2_{p,\alpha}$, where $\chi^2_{p,\alpha}$ is the upper α quantile of the χ^2_p distribution, i.e., $\mathbb{P}(\chi^2_p > \chi^2_{p,\alpha}) = \alpha$.
- The Fisherian approach is to state the result as a p-value where $p = \mathbb{P}(\chi^2_p > \zeta^2_{\text{obs}})$, and ζ^2_{obs} is the observed value of the statistic ζ^2.

The multivariate equivalent of a confidence interval is a **confidence region** and the $100(1-\alpha)\%$ confidence region for $\boldsymbol{\mu}$ is $\{\mathbf{a} : \zeta^2 \leq \chi^2_{p,\alpha}\}$. This confidence region will be the interior of an ellipse or ellipsoid.

Example

Consider again the exam marks for first year maths students in the two modules PRB and STA. The scatterplot below shows the module marks for $n = 203$ students on probability (PRB, x_1) and statistics (STA, x_2), with the sample mean vector $\begin{pmatrix} 62.1 \\ 62.7 \end{pmatrix}$ marked on as a red '+'.



The target for the module mean for a large population of students should be exactly 60 for both modules. We now conduct a hypothesis test to see if the lecturers have missed the target and made the exam too difficult. We will test the hypotheses H_0 versus H_1 at the 5% level where

$$H_0 : \boldsymbol{\mu} = \begin{pmatrix} 60 \\ 60 \end{pmatrix} \qquad \text{and} \qquad H_1 : \boldsymbol{\mu} \neq \begin{pmatrix} 60 \\ 60 \end{pmatrix}.$$

Let's assume to begin with that observations $\mathbf{x}_1, \ldots, \mathbf{x}_{203}$ are a random sample from $N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\Sigma} = \begin{pmatrix} 200 & 150 \\ 150 & 300 \end{pmatrix}$$

is assumed known.

The test statistic is

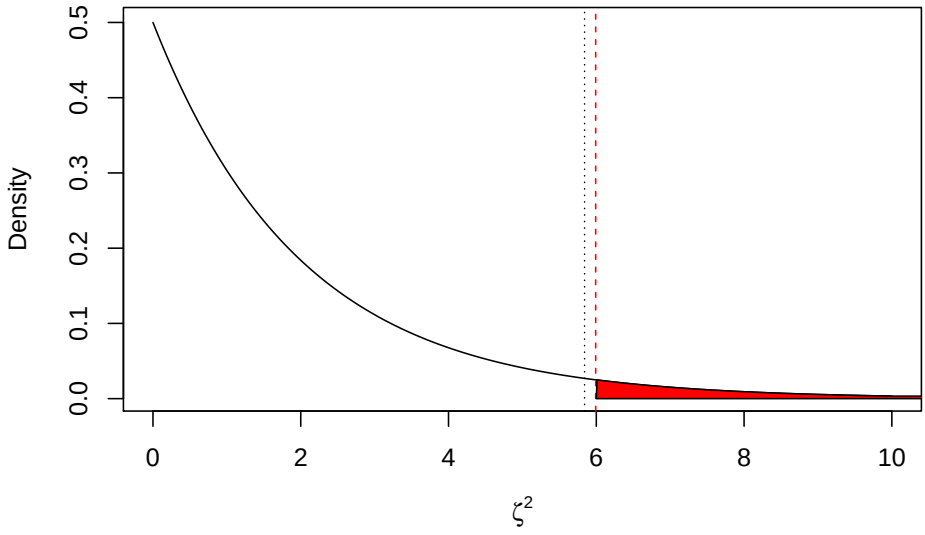
$$\begin{aligned} \zeta^2 &= 203 \begin{pmatrix} 62.1 - 60 \\ 62.7 - 60 \end{pmatrix}^\top \begin{pmatrix} 200 & 150 \\ 150 & 300 \end{pmatrix}^{-1} \begin{pmatrix} 62.1 - 60 \\ 62.7 - 60 \end{pmatrix} \\ &= 5.84 \end{aligned}$$

Under H_0, $\zeta^2 \sim \chi^2_2$. and so the critical value is $\chi^2_{2,0.05} = 5.991$.

```
qchisq(0.95, 2)
```

```
## [1] 5.991465
```

The plot below shows the density of a χ^2_2 random variable. The vertical red line shows the critical value, the vertical black line the observed value, and the shaded region shows the critical region. As $\zeta^2 < \chi^2_{p,0.05}$ we can see that we do not reject the null hypothesis at the 5% level.



The p-value is

```
1-pchisq(zeta2, 2)
```

```
##             [,1]
## [1,] 0.05389049
```

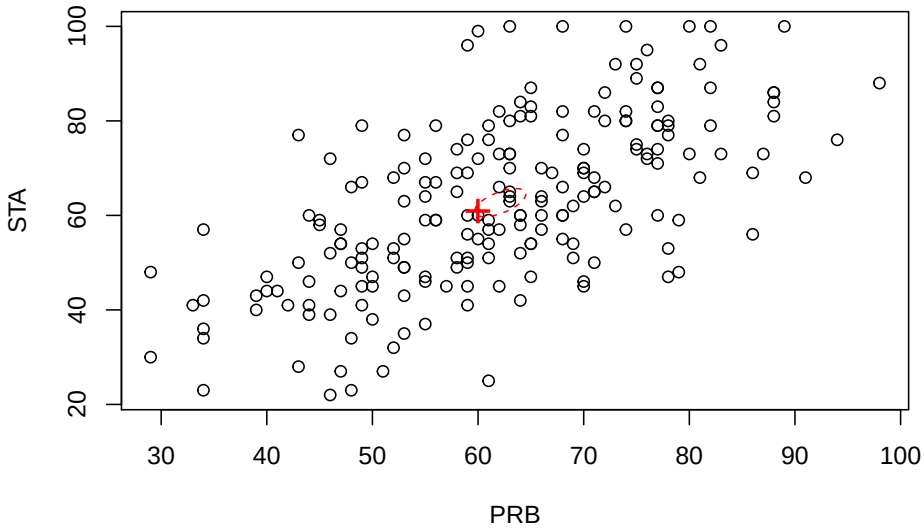
which is the area of the red shaded region.

Note that if we had conducted separate univariate hypothesis tests of $H_0 : \mu_1 = 60$ and $H_0 : \mu_2 = 60$ then the test statistics would have been:

$$\begin{aligned} z_1 &= \frac{\bar{x}_1 - \mu_1}{\sqrt{\sigma_1^2/n}} = \frac{62-60}{\sqrt{200/203}} = 2.11 \\ z_2 &= \frac{\bar{x}_2 - \mu_2}{\sqrt{\sigma_2^2/n}} = \frac{62-60}{\sqrt{300/203}} = 2.22. \end{aligned}$$

The critical value would have been $Z_{0.025} = 1.960$ and both null hypotheses would have been rejected. Therefore we see that a multivariate hypothesis can be accepted when each of univariate components is rejected and vice-versa.

The 95% confidence region is the interior of an ellipse, centred on $\bar{\mathbf{x}}$, with the angle of the major-axis governed by $\boldsymbol{\Sigma}$ (given by the eigenvectors of $\boldsymbol{\Sigma}$). We can see from the plot below that $(60, 60)^\top$, marked with a cross, lies just inside the confidence region.



7.4.2 $\boldsymbol{\Sigma}$ unknown: 1 sample

In the previous section we considered a hypothesis test of

$$H_0 : \boldsymbol{\mu} = \mathbf{a} \text{ vs } H_1 : \boldsymbol{\mu} \neq \mathbf{a}$$

based on an IID sample from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ when $\boldsymbol{\Sigma}$ was **known**. In reality, we rarely know $\boldsymbol{\Sigma}$, so we **replace it with the sample covariance matrix**, $\mathbf{S}$. Corollary 7.4 tells us that the distribution is then $F_{p,n-p}$ rather than χ^2_p as was the case when $\boldsymbol{\Sigma}$ was known.

More specifically, we use the test statistic:

$$\gamma^2 = \frac{n-p}{p}(\bar{\mathbf{x}} - \mathbf{a})^\top \mathbf{S}^{-1}(\bar{\mathbf{x}} - \mathbf{a}),$$

Corollary 7.4 tells us that when H_0 is true,

$$\gamma^2 \sim F_{p,n-p}.$$

As before, depending upon our approach we either

- (Neyman-Pearson approach) reject H_0 if $\gamma^2 > F_{p,n-p,\alpha}$, where α is the significance level.
- (Fisherian approach) compute the p-value $p = \mathbb{P}(F_{p,n-p} > \gamma^2_{obs})$.

The $100(1-\alpha)\%$ confidence region for $\boldsymbol{\mu}$ is $\{\mathbf{a} : \gamma^2 \leq F_{p,n-p,\alpha}\}$, which will again be the interior of an ellipse or ellipsoid, but the confidence region is now determined by $\mathbf{S}$ rather than $\boldsymbol{\Sigma}$.

Example continued

We return to the example with the module marks for $n = 203$ students on probability (PRB, x_1) and statistics (STA, x_2), but now we assume that $\mathbf{\Sigma}$ is unknown.

The sample mean and sample covariance matrix are

$$\bar{\mathbf{x}} = \begin{pmatrix} 62.1 \\ 62.7 \end{pmatrix} \qquad \mathbf{S} = \begin{pmatrix} 191 & 155.6 \\ 155.6 & 313.5 \end{pmatrix}$$

We conduct a hypothesis test at the 5% level of:

$$H_0 : \boldsymbol{\mu} = \begin{pmatrix} 60 \\ 60 \end{pmatrix} \qquad \text{vs.} \qquad H_1 : \boldsymbol{\mu} \neq \begin{pmatrix} 60 \\ 60 \end{pmatrix}.$$

The test statistic is

$$\begin{aligned} \gamma^2 &= \frac{203-2}{2} \begin{pmatrix} 62.1-60 \\ 62.7-60 \end{pmatrix}^\top \begin{pmatrix} 191 & 155.6 \\ 155.6 & 313.5 \end{pmatrix}^{-1} \begin{pmatrix} 62.1-60 \\ 62.7-60 \end{pmatrix} \\ &= 2.84. \end{aligned}$$

The critical value is $F_{2,201,0.05}$

```
qf(0.95, 2,201)
```

```
## [1] 3.040828
```

so $\gamma^2 < F_{p,n-p,0.05}$ and we do not reject the null hypothesis at the 5% level.

The p-value is

```
1-pf(gamma2, 2, 201)
```

```
##             [,1]
## [1,] 0.06051421
```

Thankfully, we don't need to do all the computation ourselves whenever we want to do a test, as there are several R packages that will do the work for us:

```
library(ICSNP) # you'll need to install this package the first time
HotellingsT2(X,  mu = mu)
```

```
##
##  Hotelling's one sample T2-test
##
## data:  X
## T.2 = 2.8444, df1 = 2, df2 = 201, p-value = 0.06051
## alternative hypothesis: true location is not equal to c(60,60)
```

Notice again the difference to the two univariate tests

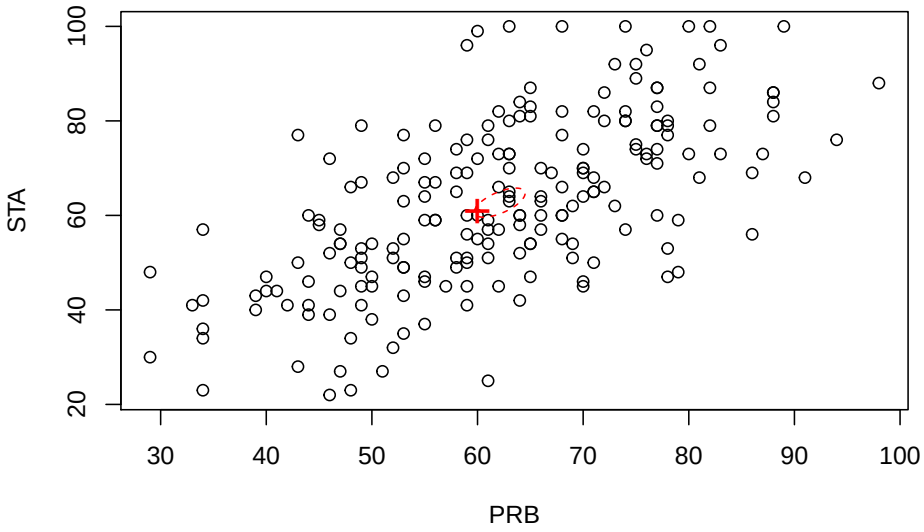
```
t.test(X[,1], mu=60)
```

```
##
##  One Sample t-test
##
## data:  X[, 1]
## t = 2.1581, df = 202, p-value = 0.0321
## alternative hypothesis: true mean is not equal to 60
## 95 percent confidence interval:
##  60.18121 64.01584
## sample estimates:
## mean of x
##  62.09852
```

```
t.test(X[,2], mu=60)
```

```
##
##  One Sample t-test
##
## data:  X[, 2]
## t = 2.1669, df = 202, p-value = 0.03141
## alternative hypothesis: true mean is not equal to 60
## 95 percent confidence interval:
##  60.24305 65.15596
## sample estimates:
## mean of x
##  62.69951
```

The 95% confidence region is the interior of an ellipse, centred on $\bar{\mathbf{x}}$, with the angle of the major-axis governed by $\mathbf{S}$. The confidence region is very slightly larger than when $\boldsymbol{\Sigma}$ was known.



7.4.3 $\boldsymbol{\Sigma}$ unknown: 2 samples

Suppose now we have data from two populations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and $\mathbf{y}_1, \ldots, \mathbf{y}_m$, and that we wish to test the difference between the two population means. As with the univariate case, there are two cases to consider:

Paired case

If $m = n$ and there exists some experimental link between $\mathbf{x}_i$ and $\mathbf{y}_i$ then we can look at the differences $\mathbf{z}_i = \mathbf{y}_i - \mathbf{x}_i$ for $i = 1, \ldots, n$. For example, $\mathbf{x}_i$ and $\mathbf{y}_i$ could be vectors of pre-treatment and post-treatment measurements, respectively, of the same variables. The crucial assumption is that the differences $\mathbf{z}_i$ are IID $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. To examine the null hypothesis of no difference between the means we would test

$$H_0 : \boldsymbol{\mu} = \mathbf{0}_p \text{ vs } H_1 : \boldsymbol{\mu} \neq \mathbf{0}_p.$$

We then base our inference on $\bar{\mathbf{z}} = \frac{1}{n}\sum_{i=1}^n \mathbf{z}_i = \bar{\mathbf{y}} - \bar{\mathbf{x}}$, and proceed exactly as in the 1 sample case, using the test in Secion 7.4.1 if $\boldsymbol{\Sigma}$ is known, or else the test in Section 7.4.2 if with $\mathbf{S} = \frac{1}{n}\sum_{i=1}^n (\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})^\top$.

Unpaired case

The unpaired case is where $\mathbf{x}_i$ and $\mathbf{y}_i$ are independent and not connected to each other. For example, in a clinical trial we may have two separate groups of patients, where one group receives a placebo and the other group receives an active treatment. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be an IID sample from $N_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and let $\mathbf{y}_1, \ldots, \mathbf{y}_m$ be an IID sample from $N_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$. In this case, we can base our inference on the following result.

Proposition 7.13. *Suppose*

$$\mathbf{x}_1, \ldots, \mathbf{x}_n \sim N_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$\mathbf{y}_1, \ldots, \mathbf{y}_m \sim N_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2).$$

Then when $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$ (i.e. when the two populations have the same distribution),

$$\frac{nm}{n+m}(\bar{\mathbf{y}} - \bar{\mathbf{x}})^\top \mathbf{S}_u^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{x}}) \sim T^2(p, n+m-2),$$

where

$$\mathbf{S}_u = \frac{n\mathbf{S}_1 + m\mathbf{S}_2}{n+m-2}$$

is the pooled unbiased variance matrix estimator and $\mathbf{S}_j$ is the sample covariance matrix for group j.

Proof. We know that $\bar{\mathbf{x}} \sim N_p\left(\boldsymbol{\mu}_1, n^{-1}\boldsymbol{\Sigma}_1\right)$ and $\bar{\mathbf{y}} \sim N_p\left(\boldsymbol{\mu}_2, m^{-1}\boldsymbol{\Sigma}_2\right)$, and $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are independent, so

$$\bar{\mathbf{y}} - \bar{\mathbf{x}} \sim N_p\left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1, \frac{1}{n}\boldsymbol{\Sigma}_1 + \frac{1}{m}\boldsymbol{\Sigma}_2\right).$$

If $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$, then $\bar{\mathbf{y}} - \bar{\mathbf{x}} \sim N_p\left(\mathbf{0}_p, \left(\frac{1}{n} + \frac{1}{m}\right)\boldsymbol{\Sigma}\right)$ and

$$\mathbf{z} = \left(\frac{1}{n} + \frac{1}{m}\right)^{-1/2}(\bar{\mathbf{y}} - \bar{\mathbf{x}}) \sim N_p(\mathbf{0}_p, \boldsymbol{\Sigma}).$$

From Proposition 7.10 we know that $n\mathbf{S}_1 \sim W_p(\Sigma_1, n-1)$ and $m\mathbf{S}_2 \sim W_p(\Sigma_2, m-1)$. Therefore when $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$,

$$\begin{aligned}\mathbf{M} = (n+m-2)\mathbf{S}_u &= (n+m-2)\frac{n\mathbf{S}_1 + m\mathbf{S}_2}{n+m-2} \\ &= n\mathbf{S}_1 + m\mathbf{S}_2 \sim W_p(\boldsymbol{\Sigma}, n+m-2)\end{aligned}$$

by Proposition 7.9, using the fact that $\mathbf{S}_1$ and $\mathbf{S}_2$ are independent.

Now $\mathbf{z}$ is independent of $\mathbf{M}$, since $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are independent of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, by Proposition 7.4. Therefore, applying Proposition 7.11 with $\mathbf{x} = \mathbf{z}$ and $\mathbf{M} = (n+m-2)\mathbf{S}_u$, we have

$$\begin{aligned}(n+m-2)\mathbf{z}^\top((n+m-2)\mathbf{S}_u)^{-1}\mathbf{z} &= \mathbf{z}^\top\mathbf{S}_u^{-1}\mathbf{z} \\ &\sim T^2(p, n+m-2)\end{aligned}$$

and

$$\begin{aligned}\mathbf{z}^\top\mathbf{S}_u^{-1}\mathbf{z} &= \left(\frac{1}{n} + \frac{1}{m}\right)^{-1/2}(\bar{\mathbf{y}} - \bar{\mathbf{x}})^\top\mathbf{S}_u^{-1}\left(\frac{1}{n} + \frac{1}{m}\right)^{-1/2}(\bar{\mathbf{y}} - \bar{\mathbf{x}}) \\ &= \left(\frac{1}{n} + \frac{1}{m}\right)^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{x}})^\top\mathbf{S}_u^{-1}(\bar{\mathbf{y}} - \bar{\mathbf{x}}).\end{aligned}$$

Finally,

$$\left(\frac{1}{n}+\frac{1}{m}\right)^{-1} = \left(\frac{m}{nm}+\frac{n}{nm}\right)^{-1} = \left(\frac{n+m}{nm}\right)^{-1} = \frac{nm}{n+m},$$

so Proposition 7.13 is proved.

□

As in the one sample case, we can convert Hotelling's two-sample T^2 statistic to the F distribution using Proposition 7.12.

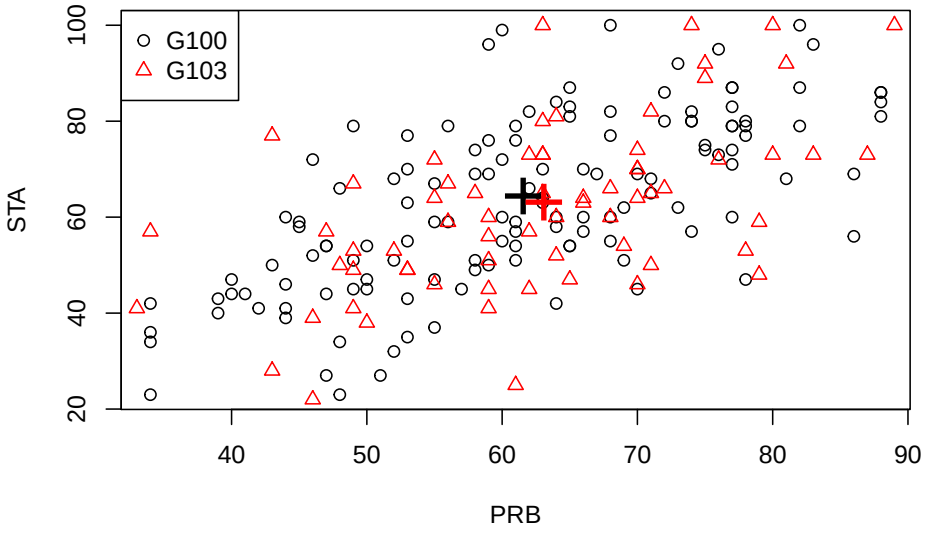
Corollary 7.5. *Using the notation of Proposition 7.13, it follows that*

$$\delta^2 = \frac{(n+m-p-1)}{(n+m-2)p}\frac{nm}{(n+m)}(\bar{\mathbf{y}}-\bar{\mathbf{x}})^\top \mathbf{S}_u^{-1}(\bar{\mathbf{y}}-\bar{\mathbf{x}}) \sim F_{p,n+m-p-1}.$$

Proof. Simply apply Proposition 7.12 to the statistic in Proposition 7.13 (replace n with $n+m-2$). □

Example continued

There are two different maths undergraduate programmes at the University of Nottingham: a 3-year (G100) and a 4-year (G103) programme. For the exam marks example, is there a significant difference between students registered on the two different programmes? Let $\mathbf{x}_1, \ldots, \mathbf{x}_{131}$ be the exam marks of the G100 students and let $\mathbf{y}_1, \ldots, \mathbf{y}_{72}$ be the exam marks of the G103 students. The data is shown below, with the sample means marked as large '+' signs.



Let $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ be the population means for G100 and G103 respectively. Our hypotheses are

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 \qquad \text{vs.} \qquad H_1 : \boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2.$$

We will assume that

$$\begin{aligned}\mathbf{x}_n, \ldots, \mathbf{x}_{131} &\sim N_2(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\\ \mathbf{y}_1, \ldots, \mathbf{y}_m &\sim N_2(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2).\end{aligned}$$

The sample summary statistics are:

$$\begin{aligned} n &= 131 & m &= 72\\ \bar{\mathbf{x}} &= \begin{pmatrix}61.6\\63.2\end{pmatrix} & \bar{\mathbf{y}} &= \begin{pmatrix}63.1\\61.9\end{pmatrix}\\ \mathbf{S}_1 &= \begin{pmatrix}180.2 & 158.5\\158.5 & 312.8\end{pmatrix} & \mathbf{S}_2 &= \begin{pmatrix}179.1 & 157.5\\157.5 & 310.8\end{pmatrix}\end{aligned}$$

The assumption $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$ does not look unreasonable given the sample covariance matrices. Using the `HotellingT2` command from the `ICSNP` package, we find

```
library(ICSNP) # you'll need to install this
HotellingsT2(G100, G103)
```

```
##
##  Hotelling's two sample T2-test
##
## data:  G100 and G103
## T.2 = 1.0696, df1 = 2, df2 = 200, p-value = 0.3451
## alternative hypothesis: true location difference is not equal to c(0,0)
```

So the test statistic was computed to be $\delta^2 = 1.06962$ and the p-value is $p = 0.345$.

The critical value for $\alpha = 0.05$ is

$$F_{2,n+m-2-1,\alpha} = F_{2,200,0.05} = 3.041.$$

```
qf(0.95,2, 200)
```

```
## [1] 3.041056
```

Therefore $\delta^2 < F_{p,n+m-p-1}$, so we do not reject the null hypothesis at the 5% level.

7.5 Exercises

1. If $M \sim W_p(\boldsymbol{\Sigma}, n)$, prove that $\mathbb{E}\mathbf{M} = n\boldsymbol{\Sigma}$.
2. If $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{a}$ is any fixed vector of dimension p, show that

$$z = \frac{\mathbf{a}^\top(\mathbf{x} - \boldsymbol{\mu})}{\sqrt{\mathbf{a}^\top \boldsymbol{\Sigma} \mathbf{a}}} \sim N(0, 1).$$

3. Prove that
$$n(\bar{\mathbf{x}} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\bar{\mathbf{x}} - \boldsymbol{\mu}) \sim \chi^2_p.$$

4. If $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are i.i.d. $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with sample mean $\bar{\mathbf{x}}$ and sample covariance matrix $\mathbf{S}$, show that $\mathbb{C}\,\mathrm{ov}(\bar{\mathbf{x}}, \mathbf{x}_i - \bar{\mathbf{x}}) = \mathbf{0}$. Hence deduce that $\bar{\mathbf{x}}$ and $\mathbf{S}$ are independent.

   Hint: If the random vector $\mathbf{x}$ is independent of the random vector $\mathbf{y}$, then $\mathbf{x}$ is also independent of any function $f(\mathbf{y})$.

5. A survey of $n = 25$ families records the head length of the first (x_1) and second (x_2) sons. The sample mean is $\bar{\mathbf{x}} = (185.72, 183.84)^\top$. Assume that the observations are sampled from $N_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} = \mathrm{diag}(100, 100)$.

   i. Conduct a hypothesis test of $\boldsymbol{\mu} = (182, 182)^\top$.
   ii. Show that the confidence region for $\boldsymbol{\mu}$ is circular. Find its centre and radius.
   iii. Repeat the hypothesis test and sketch the confidence region if we assume that
$$\boldsymbol{\Sigma} = \begin{pmatrix} 100 & 50 \\ 50 & 100 \end{pmatrix}.$$

6. Let $\mathbf{x}_1, \ldots, \mathbf{x}_{20}$ be a random sample of vectors from a $N_3(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ population where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are unknown. The sample mean and sample covariance matrix are given by
$$\bar{\mathbf{x}} = \begin{pmatrix} 0.358 \\ -1.056 \\ -1.795 \end{pmatrix} \qquad \mathbf{S} = \begin{pmatrix} 0.522 & 0.556 & -2.285 \\ 0.556 & 3.258 & -0.765 \\ -2.285 & -0.765 & 14.093 \end{pmatrix}.$$

   i. Use Hotelling's T^2 distribution to perform a significance test of the hypothesis $H_0 : \boldsymbol{\mu} = (0, -1, -1)^T$. Note that $\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ where $\boldsymbol{\Lambda} = \mathrm{diag}(14.531, 3.253, 0.090)$ and
$$\mathbf{V} = \begin{pmatrix} -0.163 & -0.121 & -0.979 \\ -0.075 & -0.988 & 0.135 \\ 0.984 & -0.095 & -0.152 \end{pmatrix}.$$

   ii. Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)^\top$. Perform separate (univariate) t-tests of the following hypotheses: $\mu_1 = 0$; $\mu_2 = -1$; $\mu_3 = -1$. Compare the results of the individual tests with the combined test based on Hotelling's T^2 distribution in (a). Comment briefly.

7. Two measurements were collected on each of 36 flea-beetles; 18 of the beetles were from a species called Chaetocnema concinna and the other 18 were from another species called Chaetocnema heikertingeri. The first variable consisted of the sum of widths (in micrometres) of the first joints of the first two tarsi ("feet"); and the second variable consisted of the corresponding sum for the second joints. It is of interest to know whether or not the population means of the two species are different.

The sample means are $\bar{\mathbf{x}}_1 = (181.50, 129.17)^\top$ and $\bar{\mathbf{x}}_2 = (205.06, 120.44)^\top$; and the sample covariance matrices are

$$\mathbf{S}_1 = \begin{pmatrix} 120.58 & 56.25 \\ 56.25 & 44.63 \end{pmatrix} \qquad \mathbf{S}_2 = \begin{pmatrix} 203.94 & 73.42 \\ 73.42 & 47.14 \end{pmatrix}.$$

Conduct a suitable hypothesis test. State your conclusion in words. What assumptions have you made in constructing the test? Do any of these assumptions seem suspect with these data?

7.6 Computer tasks

Task 1

Download the wine dataset from the UCI Machine Learning Repository

```
download.file(url="https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", des
wine <- read.csv("wine.data", header=FALSE)
colnames(wine) <- c("Type", "Alcohol", "Malic",
"Ash",
"Alcalinity",
"Magnesium",
"Phenols",
"Flavanoids",
"Nonflavanoids",
"Proanthocyanins",
"Color",
"Hue",
"Dilution",
"Proline")
```

i. Let's assume this selection of wines is a random sample from the population of all possible wines. Conduct a multivariate hypothesis test to see whether the average wine has an alcohol content different from 13 and an malic acid content different from 2?

ii. Test whether the alcohol and malic acid content are significantly different between wines of type 1 and wines of type 2.

7.6.0.0.1 Task 2

The command `rWishart` can be used to simulate from a Wishart distribution. Alternatively, to sample from $W_p(\mathbf{\Sigma}, n)$, you can sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim N_p(\mathbf{0}, \mathbf{\Sigma})$ and set $\mathbf{M} = \sum \mathbf{x}_i \mathbf{x}_i^\top$.

i. Generate 10,000 samples $M_1, \ldots,$ from a $W_2(\mathbf{\Sigma}, 10)$ distribution with $\mathbf{\Sigma} = \text{diag}(2, 1)$ using these two approaches. Compute the mean and variance of the two samples and check these accord with Proposition 7.6.

ii. Set $\mathbf{a} = (1\ 1)^\top$. Check empirically that $\mathbf{a}^\top \mathbf{M}\mathbf{a} \sim 3\chi^2_{10}$. **Hint** plot the theoretical densities on top of a histogram of the sampled quantities.

iii. Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_{10} \sim N_2(\mathbf{0}, \text{diag}(2, 1))$. Empirically check that Proposition 7.10 is true by computing the covariance matrix of a large number of such samples, and comparing this to the mean and variance of the Wishart distribution specified in the proposition.

iv. Similarly, validate Corollary 7.4 by comparing the distribution of γ^2 with a $F_{p,n-p}$ distribution.

7.6.0.0.2 Task 3

Download the exam data from Moodle.

```
library(dplyr)
load(file='exam.rda')
N<-dim(exam)[1]
```

We will now work through how to plot the confidence regions shown in the notes.

i. Firstly, let's plot a circle with equation

$$x^2 + y^2 = c^2$$

or in vector form:

$$\mathbf{x}^\top \mathbf{x} = c^2.$$

Why must x be in the range $(-c, c)$? Suppose $c = 10$. In this case you can plot a circle using the following commands

```
theta <- seq(0, 2*pi, 0.01)
x<- 10*cos(theta)
y <- 10*sin(theta)
par(pty="s")  # ensures the plot window is square
plot(x,y,type='l')
```

Explain why this works. **Note** if your plotting window is not square, your circle will look like an ellipse!

ii. Let's now plot the ellipse

$$\mathbf{x}^\top \mathbf{S}^{-1} \mathbf{x} = c^2$$

where $\mathbf{S}$ is the covariance matrix for the exam data.

We can do this by noting that $\mathbf{u} = \mathbf{S}^{-1/2}\mathbf{x}$ obeys the equation

$$\mathbf{u}^\top \mathbf{u} = c^2,$$

i.e., a circle. Thus you can plot an ellipse by using the code above to generate a circle, and then transforming it to be an ellipse. Plot the ellipse for $\mathbf{S}$ given by the sample covariance matrix of the data.

iii. What are the major and minor axes of these ellipses?

iv. Finally, we can plot the ellipse

$$(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = c^2$$

by shifting the ellipse to be centered around $\boldsymbol{\mu}$. Thus plot the 95% confidence region for the population mean $\boldsymbol{\mu}$ for the exam data. You will need to use Corollary 7.4 to determine the value of c.

Part IV: Classification and Clustering

In Part IV, we focus on different methods of classification, i.e. allocating the observations in a sample to different subsets (or groups). We distinguish between **supervised** and **unsupervised** learning methods.

In **supervised learning**, our training data consists of the measurements $\mathbf{x}_i$ on each case, and a class label y_i. Our aim is to learn the mapping from $\mathbf{x}$ to y. Linear regression is an example of a supervised learning method.

In Chapter 8, we focus on another supervised learning approach called **discriminant analysis** which aims to allocate observations to distinct groups. We are given a training set containing data with cases and their group label, and we must use this training sample to set up a suitable classification rule (classifing cases to groups). An important type of situation where discriminant analysis is used is in screening tests. Here, several variables may be measured on each individual, and we want to decide whether each individual is *negative* for some disease, in which case no further investigations are required, or *positive*, in which case further tests are required.

In **unsupervised learning**, we do not have labelled data (ie there is no y), we only have the measurements $\mathbf{x}$. The method has to find or invent its own representation of the data structure. PCA and MDS are both unsupervised learning methods. In Chapter 9, we will consider **cluster analysis**, in which we group observations into clusters (or similar subsets). Here, data labels (y variables) are not available and the number of clusters will not typically be known in advance. The idea is to form clusters in such a way that experimental units within clusters are as similar as possible, in a suitable sense, and experimental units in different clusters are as dissimilar as possible.

Chapter 8

Discriminant analysis

The videos for this chapter are available at the links below:

- Introduction
- Maximum likelihood discriminant rule
- Two populations
- Bayes discriminant rule
- Example: Iris data
- Fisher's LDA

- Iris example revisited + links to other methods

We consider the situation in which there are g populations/classes $\Pi_1, \ldots, \Pi_g$. Each subject/case belongs to precisely one population. For example

- for the iris data the populations are the 3 species {Setosa, Versicolor, Viriginica}. Each iris has a single species label.
- for the MNIST data, the populations are the digits 0 to 9. Each image represents a single digit.

Suppose we are given measurements $\mathbf{x}_i$ on each subject, and population/class label y_i. The aim of **classification** is to build a model to predict the class label y from a set of measurements $\mathbf{x}$. For our examples:

- iris: the measurements $\mathbf{x}_i \in \mathbb{R}^4$ are the sepal and petal width and lengths. The class label y_i is the species.
- mnist: $\mathbf{x}_i \in \mathbb{R}^{784}$ is a vector of pixel intensities. The class label y_i is the digit the image represents.

If $\mathbf{x} \in \mathbb{R}^p$ is a 'new' observation assumed to come from one of $\Pi_1, \ldots, \Pi_g$, the aim of discriminant analysis is to allocate $\mathbf{x}$ to one of $\Pi_1, \ldots, \Pi_g$ with **as small a probability of error as possible**.

A **discriminant rule**, d, corresponds to a partition of $\mathbb{R}^p$ into disjoint regions $\mathcal{R}_1, \ldots, \mathcal{R}_g$, where

$$\bigcup_{j=1}^{g} \mathcal{R}_j = \mathbb{R}^p, \qquad \mathcal{R}_j \cap \mathcal{R}_k = \emptyset, j \neq k.$$

The rule d is then defined by

d: allocate $\mathbf{x}$ to Π_j if and only if $\mathbf{x} \in \mathcal{R}_j$.

The way we find a rule d, is by defining discriminant functions, $\delta_j(\mathbf{x})$, for each population $j = 1, \ldots, g$. We classify $\mathbf{x}$ to population j if $\delta_j(\mathbf{x}) > \delta_i(\mathbf{x})$ for $i \neq j$. We can write this as

$$d(\mathbf{x}) = \arg\max_j \delta_j(\mathbf{x}),$$

i.e., d is a map from $\mathbb{R}^p$ to $\{1, \ldots, g\}$

Linear discriminant analysis

We will focus on discriminant functions that are **affine** functions of the data. That is they are linear projections of the data plus a constant of the form

$$\delta_j(\mathbf{x}) = \mathbf{v}_j^\top \mathbf{x} + c_j. \tag{8.1}$$

In later sections we will discuss how to choose the discriminant rules $\delta_j(\mathbf{x})$, i.e., how to choose the *parameters* $\mathbf{v}_j$ and c_j. But first we focus on what the discriminant regions look like and how to classify points given a discriminant rule.

Geometry of LDA

Proposition 8.1. *The discriminant regions $\mathcal{R}_j$ corresponding to affine discriminant functions are convex and connected.*

Proof. You will prove this in the exercises. □

Recall that the equation of a hyperplane of dimension $p-1$ in $\mathbb{R}^p$ requires us to specify a point on the plane, $\mathbf{x}_0$, and a vector from the origin that is perpendicular/orthogonal to the plane, $\mathbf{n}$. The hyperplane can then be described as

$$\mathcal{H}(\mathbf{n}, \mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^p : \mathbf{n}^\top(\mathbf{x} - \mathbf{x}_0) = 0\}$$

The orthogonal distance from the origin to the plane is

$$\frac{1}{||\mathbf{n}||}\mathbf{x}_0^\top \mathbf{n}.$$

Affine discriminant functions (8.1) divide $\mathbb{R}^p$ into regions using $(p-1)$-dimensional hyperplanes, which results in decision regions that have linear boundaries.

To see this, consider the cases where there are just two populations ($g = 2$). We classify a point $\mathbf{x}$ as population 1 if $\delta_1(\mathbf{x}) > \delta_2(\mathbf{x})$, and the decision boundary is at

$$\delta_1(\mathbf{x}) = \delta_2(\mathbf{x}).$$

If $\mathbf{x}$ is on the decision boundary then

$$(\mathbf{v}_1 - \mathbf{v}_2)^\top \mathbf{x} + c_1 - c_2 = 0.$$

This can be seen to be the equation of a hyperplane with normal vecor $\mathbf{v}_1 - \mathbf{v}_2$. The orthogonal distance of the plane from the origin is

$$\frac{c_2 - c_1}{||\mathbf{v}_1 - \mathbf{v}_2||}.$$

Example 8.1. Let's consider the case of 3 populations with data in $\mathbb{R}^2$. Suppose the discriminant functions have

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \quad \mathbf{v}_3 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

and

$$c_1 = 1, \quad c_2 = -1, \quad c_3 = 0.$$

Then

- the decision boundary between population 1 and 2 is the line with equation

  $$y = x - 2$$

  and has orthogonal distance $\sqrt{2}$ from the origin;
- the decision boundary between population 1 and 3 is the line with equation

  $$y = -\frac{1}{2}$$

  which has orthogonal distance $\frac{1}{2}$ from the origin;
- the decision boundary between population 2 and 3 is the line with equation

  $$y = -x + 1.$$

  which has orthogonal distance $\frac{1}{\sqrt{2}}$ from the origin.

If we plot these, then you can see that this results in the decision regions shown in Figure 8.1.

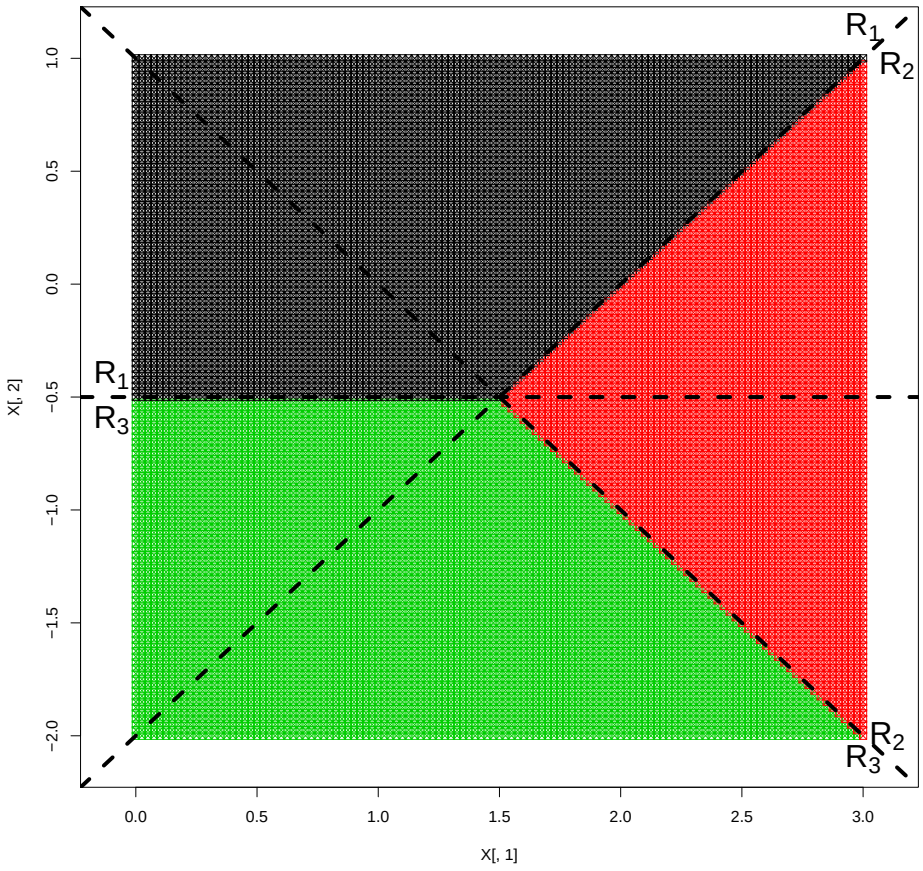


Figure 8.1: Discriminant regions for Example 1.

In the following sections we consider several different approaches to deciding upon a suitable choice of $\mathbf{v}_j$ and c_j in Equation (8.1).

1. By assuming the populations have a multivariate normal distribution. This is the focus of section 8.1, where we assume each population is equally likely. We generalise this approach in Section 8.2 to consider the situation where observations are more likely to be some from populations than from others.

2. By looking for linear projections of the data that maximize the spread between the different populations, i.e., that are best for distinguishing between different populations. This leads to **Fisher's discriminant rule** which we cover in Section 8.3, and it does not require us to make assumptions about the distribution of the underlying populations.

8.1 Maximum likelihood (ML) discriminant rule

A popular way to develop a discriminant rule is to start by assuming (or estimating) a different distribution for $\mathbf{x} \in \mathbb{R}^p$ for each population. For example, suppose that the observations in population j have a distribution with pdf $f_j(\mathbf{x})$, for $j = 1, \ldots, g$.

We will begin by supposing the different population distributions $f_1(\mathbf{x}), \ldots, f_g(\mathbf{x})$ are **known**, and in particular, that they are multivariate normal distributions.

Definition 8.1. The **maximum likelihood** (ML) discriminant rule allocates $\mathbf{x}$ to the population with the largest likelihood at $\mathbf{x}$, i.e. it allocates $\mathbf{x}$ to Π_j where

$$f_j(\mathbf{x}) = \max_{1 \leq k \leq g} f_k(\mathbf{x}).$$

We can write the discriminant rule as

$$d(\mathbf{x}) = \arg\max_k f_k(\mathbf{x}).$$

Example 8.2. Consider the univariate case with $g = 2$ where Π_1 is the $N(\mu_1, \sigma_1^2)$ distribution and Π_2 is the $N(\mu_2, \sigma_2^2)$ distribution. The ML discriminant rule allocates x to Π_1 if and only if

$$f_1(x) > f_2(x),$$

which is equivalent to

$$\frac{1}{(2\pi\sigma_1^2)^{1/2}} \exp\left(-\frac{1}{2\sigma_1^2}(x-\mu_1)^2\right) > \frac{1}{(2\pi\sigma_2^2)^{1/2}} \exp\left(-\frac{1}{2\sigma_2^2}(x-\mu_2)^2\right).$$

Collecting terms together on the left hand side (LHS) gives

$$\frac{\sigma_2}{\sigma_1}\exp\left(-\frac{1}{2\sigma_1^2}(x-\mu_1)^2+\frac{1}{2\sigma_2^2}(x-\mu_2)^2\right)>1 \tag{8.2}$$

$$\iff \quad \log\left(\frac{\sigma_2}{\sigma_1}\right)-\frac{1}{2\sigma_1^2}(x-\mu_1)^2+\frac{1}{2\sigma_2^2}(x-\mu_2)^2>0 \tag{8.3}$$

$$\iff \quad x^2\left(\frac{1}{\sigma_2^2}-\frac{1}{\sigma_1^2}\right)+x\left(\frac{2\mu_1}{\sigma_1^2}-\frac{2\mu_2}{\sigma_2^2}\right)+\frac{\mu_2^2}{\sigma_2^2}-\frac{\mu_1^2}{\sigma_1^2}+2\log\frac{\sigma_2}{\sigma_1}>0 \tag{8.4}$$

Suppose, for example, that $\mu_1=\sigma_1=1$ and $\mu_2=\sigma_2=2$, then this reduces to the quadratic expression

$$-\frac{3}{4}x^2+x+2\log 2>0.$$

Suppose that our new observation is $x=0$, say. Then the LHS is $2\log 2$ which is greater than zero and so we would allocate x to population 1.

Using the quadratic equation formula we find that $f_1(x)=f_2(x)$ when

$$x=\frac{-1\pm\sqrt{1+6\log 2}}{-3/2}=\frac{2}{3}\pm\frac{2}{3}\sqrt{1+6\log 2},$$

i.e. at $x=-0.85$ and $x=2.18$. Our discriminant rule is thus to allocate x to Π_1 if $-0.85<x<2.18$ and to allocate it to Π_2 otherwise. This is illustrated in Figure 8.2.

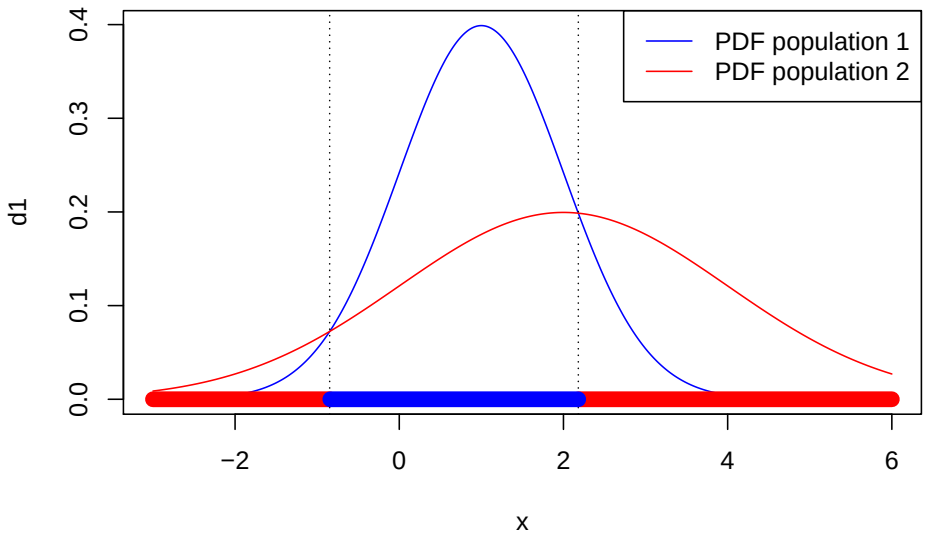


Figure 8.2: Discriminant rule for the two Gaussians example.

Note that this has not resulted in connected convex discriminant regions $\mathcal{R}_i$. This is because our discriminant functions were not linear functions of $\mathbf{x}$ - thus we did not find a linear discriminant rule.

Note also that if $\sigma_1 = \sigma_2$ then the x^2 term in Equation (8.4) cancels, and we are left with a linear discriminant rule. For example, if $\sigma_2 = 1$ with the other parameters as before, then we classify x to population 1 if

$$2x\,(\mu_1 - \mu_2) + \mu_2^2 - \mu_1^2 = -2x + 3 > 0.$$

i.e., if $x < \frac{3}{2}$. In this case, we do get discriminant regions $\mathcal{R}_j$ which are connected and convex.

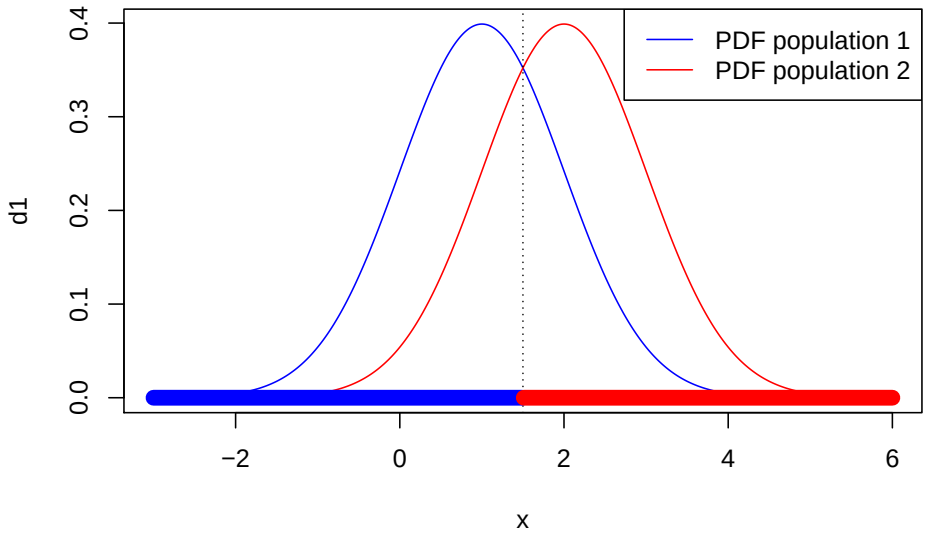


Figure 8.3: Discriminant rule for the two Gaussians example when sigma_2=1

8.1.1 Multivariate normal populations

Now we consider the case of g multivariate normal populations. We shall assume that for population k

$$\mathbf{x} \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

i.e., we allow the mean of each population to vary, but have assumed a common covariance matrix between groups. We call the $\boldsymbol{\mu}_k$ the **population means** or **centroids**.

Proposition 8.2. *If cases in population Π_k have a $N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ distribution, then the ML discriminant rule is*

$$d(\mathbf{x}) = \arg\min_k (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k).$$

Equivalently, if $\delta_k(\mathbf{x}) = 2\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k$. *Then*

$$d(\mathbf{x}) = \arg\max \delta_k(\mathbf{x}).$$

I.e. this is a linear discriminant rule.

Proof. The kth likelihood is

$$f_k(\mathbf{x}) = |2\pi\boldsymbol{\Sigma}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right). \tag{8.5}$$

This is maximised when the exponent is minimised, due to the minus sign in the exponent and because $\boldsymbol{\Sigma}$ is positive definite.

$$\begin{aligned}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) &= \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \\ &= \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - \delta_k(\mathbf{x})\end{aligned}$$

Thus,

$$\arg\min_k (\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) = \arg\max_k \delta_k(\mathbf{x})$$

as $\mathbf{x}^\top \boldsymbol{\Sigma}\mathbf{x}$ does not depend on k. □

8.1.2 The sample ML discriminant rule

To use the ML discriminant rule, we need to know the model parameters for each group, $\boldsymbol{\mu}_k$, as well as the common covariance matrix $\boldsymbol{\Sigma}$. We will usually not know these parameters, and instead must estimate them from **training** data. We then substitute these estimates into the discriminant rule. Training data typically consists of samples $\mathbf{x}_{1,k}, \ldots, \mathbf{x}_{n_k,k}$ known to be from population Π_k, where n_k is the number of observations from population Π_k.

We estimate the unknown population means by the sample mean for each population

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} \mathbf{x}_{i,k}.$$

To estimate the shared covariance matrix, $\boldsymbol{\Sigma}$, first calculate the sample covariance matrix for the kth group:

$$\mathbf{S}_j = \frac{1}{n_j - 1}\sum_{i=1}^{n_j} (\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j)^\top$$

Then

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-g}\sum_{k=1}^{g} n_k \mathbf{S}_k \tag{8.6}$$

is an unbiased estimate of $\mathbf{\Sigma}$ where $n = n_1 + n_2 + \ldots + n_g$. Note that this is not the same as the total covariance matrix (i.e. ignoring the class labels).

The sample ML discriminant rule is then defined by substituting these estimates into 8.2.

8.1.3 Two populations

Corollary 8.1. *When $g = 2$, the rule allocates $\mathbf{x}$ to Π_1 if and only if*

$$\mathbf{a}^\top(\mathbf{x} - \mathbf{h}) > 0,$$

where $\mathbf{a} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ and $\mathbf{h} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)$.

Proof. From Proposition 8.2 we know that we classify to population 1 if and only if

$$\delta_1(\mathbf{x}) > \delta_2(\mathbf{x}),$$

i.e., if

$$2\boldsymbol{\mu}_1^\top \mathbf{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\mu}_1\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 > 2\boldsymbol{\mu}_2^\top \mathbf{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\mu}_2\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2$$

Rearranging we see this is true if and only if

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \mathbf{\Sigma}^{-1}\mathbf{x} > \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2).$$

Using the expressions for $\mathbf{a}$ and $\mathbf{h}$ gives

$$\mathbf{a}^\top\mathbf{x} > \mathbf{a}^\top\mathbf{h}$$

from which the desired result follows. □

If we think about the situation where $\mathbf{\Sigma} = \mathbf{I}$, then we can make sense of this rule geometrically. If the variance of the two populations is the identity matrix, then we can simply classify to the nearest population mean/centroid, and the decision boundary is thus the perpendicular bisector of the two centroids. Moreover,

- $\mathbf{a} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ is the vector between the two population centroids, and thus will be perpendicular to the decision boundary.
- $\mathbf{h} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)$ is the midpoint of the two population centroids, and thus $\mathbf{h}$ will lie on the decision boundary.
- An equation for the decision boundary is $\mathbf{a}^\top(\mathbf{x} - \mathbf{h}) = 0$.
- For a general point $\mathbf{x} \in \mathbb{R}^p$, the vector $\mathbf{x} - \mathbf{h}$ is a vector from the decision boundary (i.e. from $\mathbf{h}$) to $\mathbf{x}$.
- Thinking of the scalar product, we can see that $\mathbf{a}^\top(\mathbf{x} - \mathbf{h})$ is proportional to the cosine of the angle between $\mathbf{a}$ and $\mathbf{x} - \mathbf{h}$. The point $\mathbf{x}$ will be closer to $\boldsymbol{\mu}_1$ than $\boldsymbol{\mu}_2$ if the angle between $\mathbf{a}$ and $\mathbf{x} - \mathbf{h}$ is between $-90°$ and $90°$, or equivalently, if the cosine of the angle is greater than 0.

- Thus we classify $\mathbf{x}$ to population 1 if $\mathbf{a}^\top(\mathbf{x}-\mathbf{h}) > 0$, and to population 2 if $\mathbf{a}^\top(\mathbf{x}-\mathbf{h}) < 0$.

This situation is illustrated in Figure **??**.

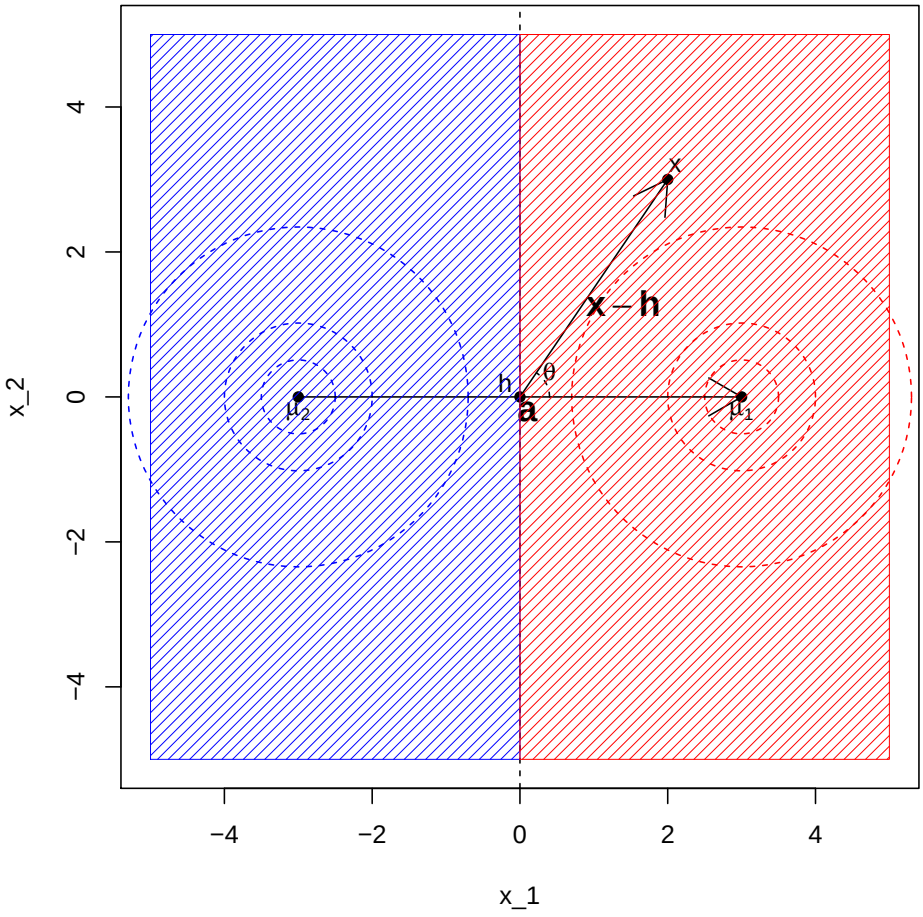
If we have more than 2 populations, then for $\mathbf{\Sigma} = \mathbf{I}$, the decision boundaries are the perpendicular bisectors between the population centroids (the $\boldsymbol{\mu}_i$) and we simply classify to the nearest centroid.

When $\mathbf{\Sigma} \neq \mathbf{I}$, we think of $\mathbf{\Sigma}$ as distorting space. Instead of measuring distance using the Euclidean distance, we instead adjust distances to account for $\mathbf{\Sigma}$. The decision boundaries are then no longer the perpendicular bisectors of the centroids.

8.1.3.0.1 Example 2

Consider the bivariate case ($p = 2$) with $g = 2$ groups, where Π_1 is the $N_2(\boldsymbol{\mu}_1, \mathbf{I}_2)$ distribution and Π_2 is the $N_2(\boldsymbol{\mu}_2, \mathbf{I}_2)$ distribution. Suppose $\boldsymbol{\mu}_1 = \begin{pmatrix} c \\ 0 \end{pmatrix}$ and $\boldsymbol{\mu}_2 = \begin{pmatrix} -c \\ 0 \end{pmatrix}$ for some constant $c > 0$. Here, $\mathbf{a} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = \begin{pmatrix} 2c \\ 0 \end{pmatrix}$ and $\mathbf{h} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

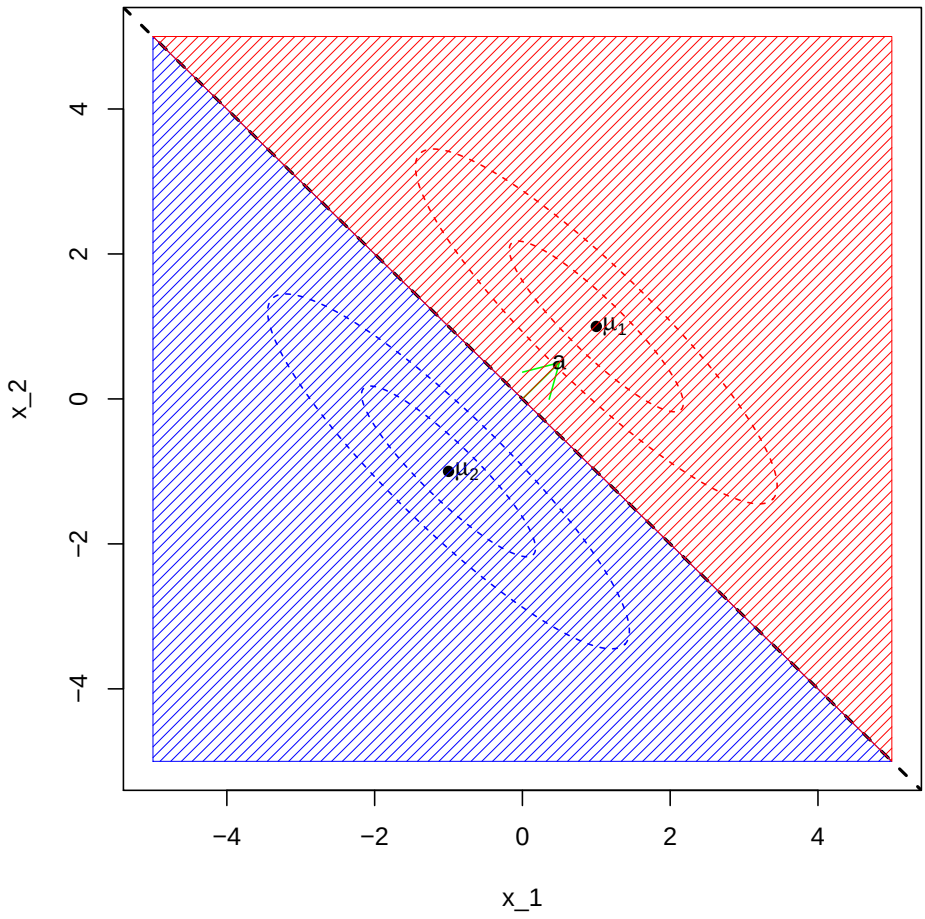
The ML discriminant rule allocates $\mathbf{x}$ to Π_1 if $\mathbf{a}^\top(\mathbf{x}-\mathbf{h}) = \mathbf{a}^\top\mathbf{x} > 0$. If we write $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ then $\mathbf{a}^\top\mathbf{x} = 2cx_1$, which is greater than zero if $x_1 > 0$. Hence we allocate $\mathbf{x}$ to Π_1 if $x_1 > 0$ and allocate $\mathbf{x}$ to Π_2 if $x_1 \leq 0$.



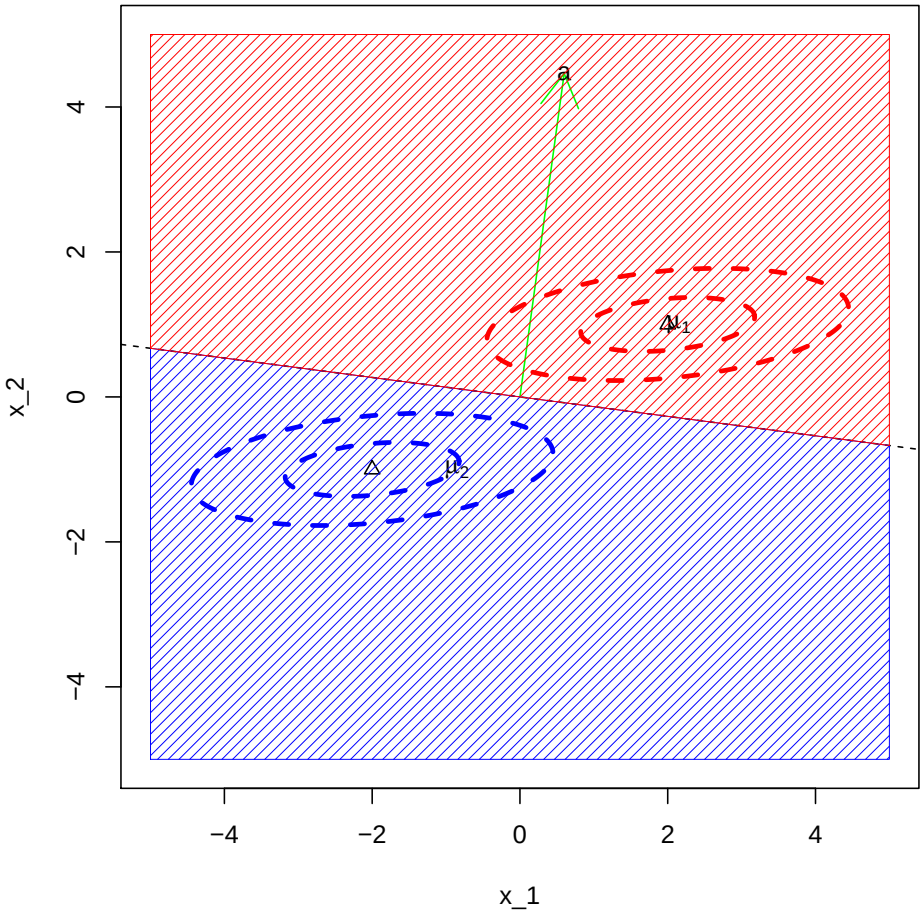
8.1.3.0.2 Example 3

Let's now generalise the previous example, making no assumptions about $\mathbf{\Sigma}$, but still assuming $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_2$. If we write $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ and $\mathbf{h} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) = \mathbf{0}$. Then the ML discriminant rule allocates $\mathbf{x}$ to Π_1 if $\mathbf{a}^\top \mathbf{x} > 0$. If we write $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ then the boundary separating $\mathcal{R}_1$ and $\mathcal{R}_2$ is given by $\mathbf{a}^\top \mathbf{x} = \begin{pmatrix} a_1 & a_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = a_1 x + a_2 y = 0$, i.e. $y = -\frac{a_1}{a_2} x$. This is a straight line through the origin with gradient $-a_1/a_2$.

For $\boldsymbol{\mu}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\mathbf{\Sigma} = \begin{pmatrix} 1 & -0.9 \\ -0.9 & 1 \end{pmatrix}$ we find $\mathbf{a} = \begin{pmatrix} 20 \\ 20 \end{pmatrix}$, which gives the line $y = -1x$.



If the variance of the y component is very small compared to the variance of the x component, then we begin to classify solely on the basis of y. For example, if $\boldsymbol{\mu}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ and $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.09 \\ 0.09 & 0.1 \end{pmatrix}$ we find $\mathbf{a} = \begin{pmatrix} 2.39 \\ 17.8 \end{pmatrix}$, which gives the line $y = -0.13x$. I.e., a line that is getting close to being horizontal.



8.1.4 More than two populations

When $g > 2$, the boundaries for the ML rule will be piece-wise linear. In the exercises you will look at an example with 3 populations in two dimensions.

8.2 Bayes discriminant rule

In the previous section, we implicitly assumed that each subject is equally likely to be from any of the g populations. This is the simplest case but is an unrealistic assumption in practice.

For example, suppose we want to classify photos on the internet as either being a photo of *Bill Evans* or *not Bill Evans*. Most photos on the internet are not of Bill Evans, and so we want to take this into account in the classifier, i.e., we want to take the base rate of occurence of each population into account.

Suppose *a priori* that the probability an observation is from population k is π_k,

with

$$\sum_{k=1}^{g} \pi_k = 1.$$

Then given a probability model $f_k(\mathbf{x})$ for observations $\mathbf{x}$ from population k, our posterior probability for observation $\mathbf{x}$ being from population k is

$$\mathbb{P}(y = k \mid \mathbf{x}) = \frac{f_k(\mathbf{x})\pi_k}{\sum_{j=1}^{g} f_j(\mathbf{x})\pi_j}$$

by Bayes theorem.

Definition 8.2. The **Bayes classifier** assigns $\mathbf{x}$ to the population for which the posterior probability is highest:

$$d^{Bayes}(\mathbf{x}) = \arg\max_k \mathbb{P}(y = k \mid \mathbf{x}).$$

As before, if we assume each population has a multivariate normal distribution, then this simplifies.

Proposition 8.3. *If cases in population Π_k have a $N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ distribution, then the **Bayes** discriminant rule is*

$$d(\mathbf{x}) = \arg\min_k \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - \log \pi_k.$$

Equivalently, if

$$\delta_k(\mathbf{x}) = \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

then

$$d(\mathbf{x}) = \arg\max \delta_k(\mathbf{x}).$$

I.e. this is a linear discriminant rule.

Proof. See the exercises. □

Note that if $\pi_k = \frac{1}{g}$ for all k, then we can ignore π_k in the maximization, and the Bayes discriminant rule is exactly the same as the ML discriminant rule.

For two populations, the Bayes analogue of Corollary 8.1 is that we classify $\mathbf{x}$ to population 1 if

$$\mathbf{a}^\top(\mathbf{x} - \mathbf{h}) > \log \frac{\pi_2}{\pi_1}.$$

The effect of the prior probabilities on the populations is to shift the decision boundary to be further away from the more likely class. For example, if in Example 2 we had $\pi_1 = 0.9$ and $\pi_2 = 0.1$ we get the decision boundary at $x_1 = -2.19722$ as shown in Figure 8.4.

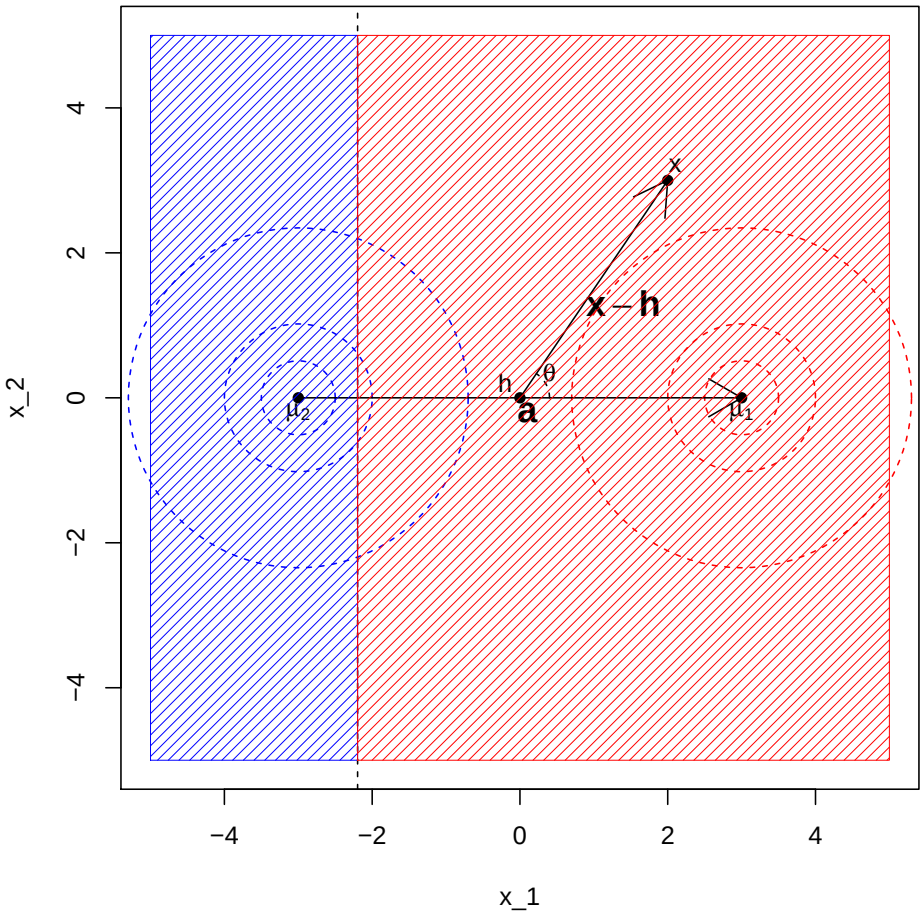


Figure 8.4: LDA when the covariance matrix is the identity, with class prior pi1=0.9

In practice, we may not know the population probabilities π_k. If so, we can estimate them from the data using

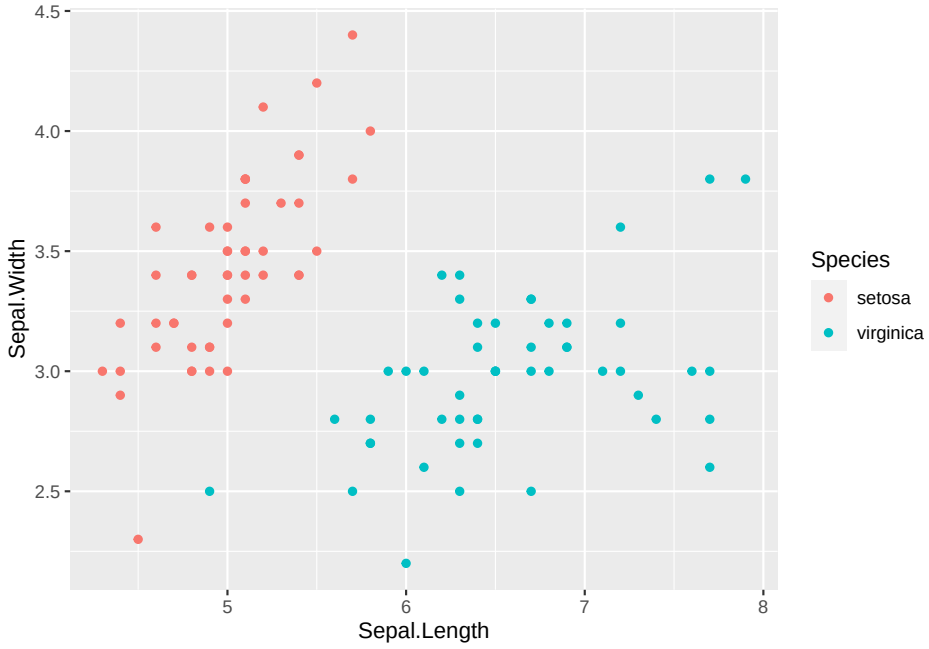
$$\hat{\pi}_k = \frac{n_k}{n}$$

and substitute $\hat{\pi}_k$ for π_k (as well as substituting $\hat{\boldsymbol{\mu}}_k$, $\widehat{\boldsymbol{\Sigma}}$ etc).

The `lda` command from the `MASS` library uses the Bayes discriminant rule, but gives you the option of setting π_k if known, but otherwise estimates the class probabilities.

8.2.1 Example: LDA using the Iris data

Let's consider doing LDA with the iris data. To begin with, lets use just the setosa and virginica species so that we only have $g = 2$ populations, Π_{setosa} and $\Pi_{virginica}$. We will also just use the sepal measurements so that $p = 2$. If we plot the data, we can see that the two populations should be easy to classify using just these two measurements:



The sample means and variances for each group are

$$\hat{\boldsymbol{\mu}}_s = \begin{pmatrix} 5.01 \\ 3.43 \end{pmatrix} \qquad \hat{\boldsymbol{\mu}}_v = \begin{pmatrix} 6.59 \\ 2.97 \end{pmatrix}$$

$$\mathbf{S}_s = \begin{pmatrix} 0.124 & 0.0992 \\ 0.0992 & 0.144 \end{pmatrix} \qquad \mathbf{S}_v = \begin{pmatrix} 0.404 & 0.0938 \\ 0.0938 & 0.104 \end{pmatrix}$$

where the s subscript gives the values for setosa, and v for virginica.

We have data on $n = 50$ flowers in each population. Hence,

$$\begin{aligned}
\widehat{\boldsymbol{\Sigma}} &= \frac{1}{50+50-2}(50\mathbf{S}_s + 50\mathbf{S}_v) = \begin{pmatrix} 0.27 & 0.0985 \\ 0.0985 & 0.126 \end{pmatrix}, \\
\hat{\boldsymbol{\mu}}_s - \hat{\boldsymbol{\mu}}_v &= \begin{pmatrix} -1.58 \\ 0.454 \end{pmatrix}, \\
\hat{\mathbf{h}} &= \frac{1}{2}(\hat{\boldsymbol{\mu}}_s + \hat{\boldsymbol{\mu}}_v) = \begin{pmatrix} 5.797 \\ 3.201 \end{pmatrix},
\end{aligned}$$

and

$$\hat{\mathbf{a}} = \widehat{\boldsymbol{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_s - \hat{\boldsymbol{\mu}}_v) = \begin{pmatrix} 5.18 & -4.04 \\ -4.04 & 11.1 \end{pmatrix} \begin{pmatrix} -1.58 \\ 0.454 \end{pmatrix} = \begin{pmatrix} -10.031 \\ 11.407 \end{pmatrix}.$$

We can find these values in R as follows:

```
library(dplyr)
# select two measurements and two species
iris3 <- iris %>% filter(Species != "versicolor") %>%
  dplyr::select(Sepal.Length, Sepal.Width, Species)

S_vir = iris3 %>% filter(Species=="virginica") %>%
  select(Sepal.Length, Sepal.Width) %>%
  cov
S_set = iris3 %>% filter(Species=="setosa") %>%
  select(Sepal.Length, Sepal.Width) %>%
  cov
S_u = (50*S_vir+50*S_set)/98


# Or more easily using a package
library(vcvComp)
S_u = cov.W(iris3[,1:2], iris3[,3])
```

The sample ML discriminant rule allocates a new observation $\mathbf{x} = (x_1, x_2)^\top$ to Π_{setosa} if and only if

$$\hat{\mathbf{a}}^\top(\mathbf{x} - \hat{\mathbf{h}}) = \begin{pmatrix} -10.031 & 11.407 \end{pmatrix} \begin{pmatrix} x_1 - 5.797 \\ x_2 - 3.201 \end{pmatrix} > 0.$$

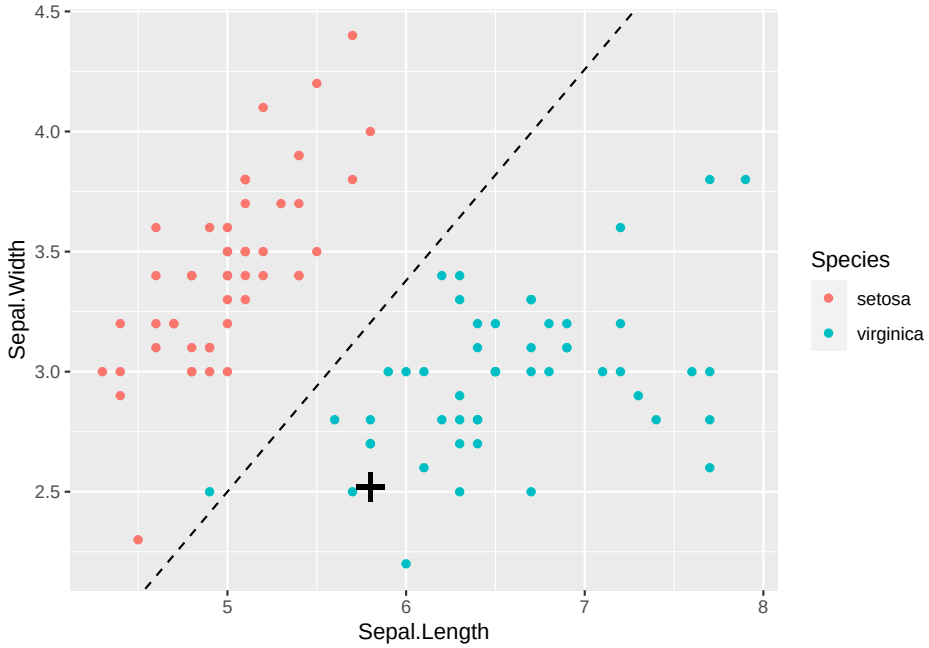
If we draw on the line defined by

$$\mathbf{a}^\top(\mathbf{x} - \mathbf{h}) = 0$$

which can be written as

$$x_2 = \frac{1}{a_2}(\mathbf{a}^\top\mathbf{h} - a_1 x_1) = -1.8963517 - -0.8793085 x_1$$

we can see this line clearly separates the two species of iris.



If a new iris had a sepal length of 5.8 and a sepal width of 2.5 (marked as a black cross on the plot above) then

$$\hat{\mathbf{a}}^\top(\mathbf{x} - \hat{\mathbf{h}}) = \begin{pmatrix} -10.031 & 11.407 \end{pmatrix} \begin{pmatrix} 5.8 - 5.797 \\ 2.5 - 3.201 \end{pmatrix} = -8.0267032 < 0,$$

and so we would allocate this iris to the virginia population, $\Pi_{virginica}$.

As you would expect, there is an R command to do this for us: `lda` (for Linear Discriminant Analysis). This is in the `MASS` R package.

```
library(MASS)
iris.lda1 <- lda(Species ~ ., iris3)

# lda(Species ~ Sepal.Length+Sepal.Width, iris3)
  #  does the same thing
iris.lda1
```

```
## Call:
## lda(Species ~ ., data = iris3)
##
## Prior probabilities of groups:
##    setosa virginica
##       0.5       0.5
##
```

```
## Group means:
##           Sepal.Length Sepal.Width
## setosa           5.006       3.428
## virginica        6.588       2.974
##
## Coefficients of linear discriminants:
##                     LD1
## Sepal.Length  2.208596
## Sepal.Width  -2.511742
```

You can see that this has printed the group means, and the lda coefficients. The coefficients are different to the ones we computed for $\mathbf{a}$, but have the same ratio, which is all that matters (because the discriminant planes are perpendicular to $\mathbf{a}$ - multiplying $\mathbf{a}$ by a constant does not change the plane).

The output also includes estimates of the prior probabilities for the two species, which are estimated to be

$$\hat{\pi}_s = \frac{1}{2}, \quad \hat{\pi}_v = \frac{1}{2}$$

as there were 50 irises from each population in the training data.

We can use the `predict` command to predict the species for a new observation $\mathbf{x}$. Using the example from before of an iris with a sepal length of 5.8 and a sepal width of 2.5 we find the Bayes classifier predicts this is a virginica iris.

```
irisnew = data.frame(Sepal.Length=5.8, Sepal.Width=2.5)
predict(iris.lda1,irisnew)
```

```
## $class
## [1] virginica
## Levels: setosa versicolor virginica
##
## $posterior
##         setosa virginica
## 1 0.0002771946 0.9997228
##
## $x
##        LD1
## 1 1.767357
```

Note that we are also given estimates of the probability that $\mathbf{x}$ lies in each of the populations. These are computed using

$$\frac{f_k(\mathbf{x})\pi_k}{\sum_{j=1}^{g} f_j(\mathbf{x})\pi_j}$$

with the parameter estimates substituted for the unknown parameters.

Using the entire dataset

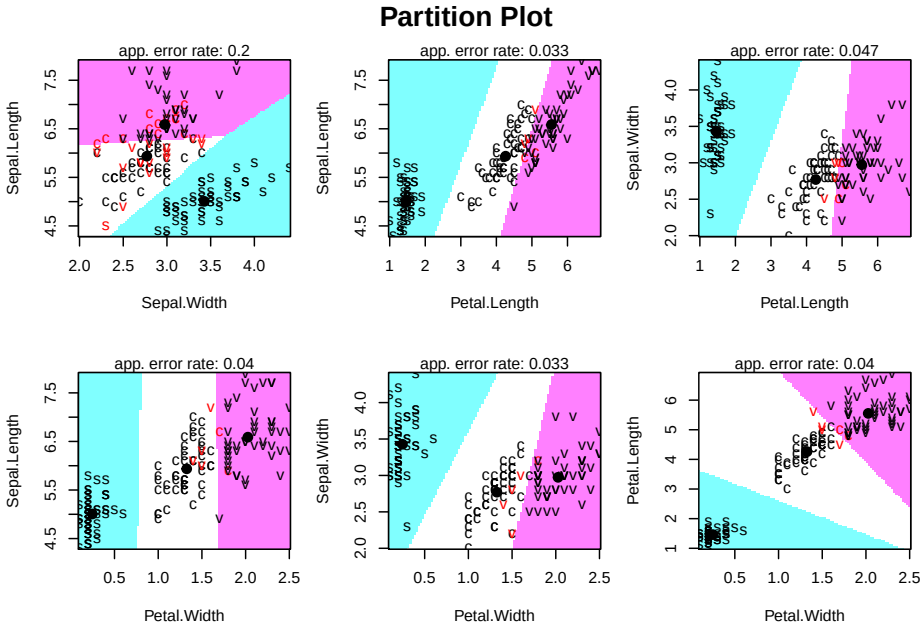
Let's now use all 150 observations (with three populations: setosa, virginica, versicolor), and all four measurements.

```
lda.iris <- lda(Species ~ .,iris)
lda.iris
```

```
## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8293776  0.02410215
## Sepal.Width   1.5344731  2.16452123
## Petal.Length -2.2012117 -0.93192121
## Petal.Width  -2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

We will delay discussion of what the reported coefficients mean, and how to visualize LDA, until after we have seen Fisher's linear discriminant analysis in the next section. But we can use the `klaR` package to visualize a little of what LDA does. The `partimat` command provides an array of figures that show the discriminant regions based on every combination of two variables (i.e., it does LDA for each pair of variables).

```
library(klaR)
library(plyr)
Species_new= revalue(iris$Species, c("versicolor"="color"))

partimat(Species_new ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
         data=iris, method="lda")
```



Note that I renamed the *versicolor* species as *color* so that the different species appear more clearly in the plots.

8.2.2 Quadratic Discriminant Analysis (QDA)

We have seen that in the case where the populations all share a common covariance matrix, $\boldsymbol{\Sigma}$, that the decision boundaries are linear (i.e., hyperplanes). We also saw in example 8.2 in a one-dimensional example that when the two populations had different variances we have a quadratic decision boundary.

In general, if we allow the variances to differ between populations, so that we model population k as

$$\mathbf{x} \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

then we get the Bayes discriminant rule

$$d(\mathbf{x}) = \arg\max_k \left(-\frac{1}{2} \log |\boldsymbol{\Sigma}_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \log \pi_k \right).$$

We can no ignore the quadratic term in $\mathbf{x}$ as it depends upon the population indicator k. We also have to include the determinant of $\boldsymbol{\Sigma}_k$. Thus in this case we get a quadratic decision boundary rather than a linear one.

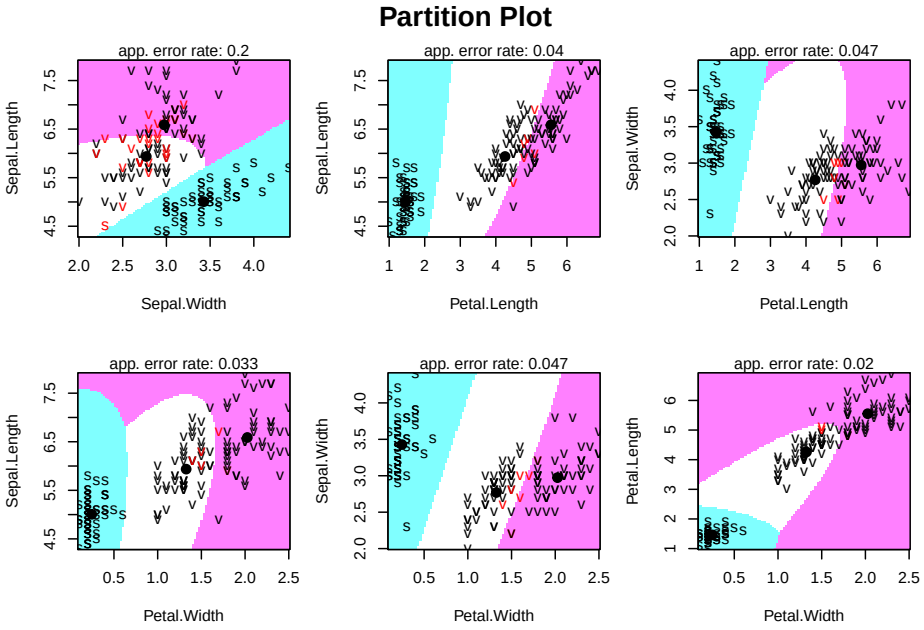
8.2.2.1 Iris example continued

The `qda` function in the `MASS` package implements quadratic discriminant analysis.

```
qda.iris <- qda(Species_new ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
```

We can again use the `partimat` command to look at the decision boundaries if we do QDA for each pair of variables in turn.

```
partimat(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,          da
```



8.2.3 Prediction accuracy

To assess the predictive accuracy of our discriminant rules, we often split the data into two parts, a **training set**, and a **test set**. Let's do this randomly by chosing 100 of the irises to be in our training set, and use the other 50 irises in the test set. We will then train the classifiers on the training set, and predict the species labels for test dataset. We can then count how many predictions were correct.

```
set.seed(2)
test.ind <- sample(150, size=50)
iris.test <- iris[test.ind,]
iris.train <- iris[-test.ind,]

lda1 <- lda(Species~., iris.train)
qda1 <- qda(Species~., iris.train)
test.ldapred <- predict(lda1, iris.test)
test.qdapred <- predict(qda1, iris.test)
(lda.accuracy <- sum(test.ldapred$class==iris.test$Species))/50
```

```
## [1] 0.98
```

```
(qda.accuracy <- sum(test.qdapred$class==iris.test$Species))/50
```

```
## [1] 0.98
```

So in this case we find that LDA and QDA both have an accuracy of 4900%, which is the result of a single wrong prediction. We can see more clearly what has gone wrong by using the `table` command.

```
table(iris.test$Species, test.ldapred$class)
```

```
##
##              setosa versicolor virginica
##   setosa         17          0         0
##   versicolor      0         15         0
##   virginica       0          1        17
```

These tables of results are known as **confusion matrices**.

So in this case one of the virginica irises was predicted incorrectly to be versicolor.

Here we randomly split the data into test and training sets. With small datasets such as this ($n = 150$), we instead sometimes use cross validation to assess the accuracy of each method. To do this, we split the data up into test and training sets multiple times randomly, and assess the prediction error on each random split.

8.3 Fisher's linear discriminant rule

Thus far we have assumed that observations from population Π_j have a $N_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma})$ distribution, and then used the MVN log-likelihood to derive the discriminant functions $\delta_j(\mathbf{x})$. The famous statistician R. A. Fisher took an alternative approach and looked for a linear discriminant functions without assuming any particular distribution for each population Π_j.

This way of thinking leads to a form of dimension reduction. We find a projection of the data into a lower dimensional space that is optimal for classifying the data into the different populations.

Variance decomposition

Suppose we have a training sample $\mathbf{x}_{1,j}, \ldots, \mathbf{x}_{n_j,j}$ from Π_j for $j = 1, \ldots, g$. Fisher's approach starts by splitting the total covariance matrix of the data (i.e. ignoring class labels) into two parts.

$$
\begin{aligned}
n\mathbf{S} = \mathbf{X}^\top \mathbf{H}\mathbf{X} &= \sum_{j=1}^{g}\sum_{i=1}^{n_j}(\mathbf{x}_{i,j} - \bar{\mathbf{x}})(\mathbf{x}_{i,j} - \bar{\mathbf{x}})^\top \\
&= \sum_{j=1}^{g}\sum_{i=1}^{n_j}(\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j + \hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})(\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j + \hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})^\top \\
&= \sum_{j=1}^{g}\sum_{i=1}^{n_j}(\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_{i,j} - \hat{\boldsymbol{\mu}}_j)^\top + \sum_{j=1}^{g} n_j(\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})(\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})^\top \\
&= n\mathbf{W} + n\mathbf{B}
\end{aligned}
$$

where $\hat{\boldsymbol{\mu}}_j = \frac{1}{n_j}\sum \mathbf{x}_{i,j} = \bar{\mathbf{x}}_{+,j}$ is the sample mean of the jth group, $\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{g}\sum_{i=1}^{n_j}\mathbf{x}_{ij}$ is the overall mean, and $n = \sum_{j=1}^{g} n_j$.

This has split the total covariance matrix into a **within-class** covariance matrix

$$\mathbf{W} = \frac{1}{n}\sum_{j=1}^{g}\sum_{i=1}^{n_j}(\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_j)^\top = \frac{1}{n}\sum_{j=1}^{g} n_j\mathbf{S}_j$$

and a **between-class** covariance matrix

$$\mathbf{B} = \frac{1}{n}\sum_{j=1}^{g} n_j(\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})(\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})^\top$$

i.e.

$$\mathbf{S} = \mathbf{W} + \mathbf{B}.$$

1. $\mathbf{W}$ is an estimator of $\boldsymbol{\Sigma}$, the shared covariance matrix in the MVN distributions for each population (c.f. Equation. (8.6)).

2. If $\mathbf{M}$ is a $n \times p$ matrix of estimated class centroids for each observation

$$\mathbf{M} = \begin{pmatrix} - & \hat{\boldsymbol{\mu}}_1 & - \\ & \vdots & \\ - & \hat{\boldsymbol{\mu}}_1 & - \\ - & \hat{\boldsymbol{\mu}}_2 & - \\ & \vdots & \\ - & \hat{\boldsymbol{\mu}}_g & - \\ & \vdots & \\ - & \hat{\boldsymbol{\mu}}_g & - \end{pmatrix}$$

   then $\mathbf{B} = \frac{1}{n}\mathbf{M}^\top\mathbf{H}\mathbf{M}$ is the covariance matrix of $\mathbf{M}$.

Fisher's criterion

Fisher's approach was to find a projection of the data $z_i = \mathbf{a}^\top \mathbf{x}_i$ or

$$\mathbf{z} = \mathbf{X}\mathbf{a}$$

in vector form, that maximizes the between-class variance relative to the within-class variance.

Using the variance decomposition from above, we can see that the total variance of $\mathbf{z}$ is

$$\begin{aligned}
\mathbb{V}\mathrm{ar}(z_i) &= \mathbb{V}\mathrm{ar}(\mathbf{a}^\top \mathbf{x}_i) \\
&= \mathbf{a}^\top \mathbb{V}\mathrm{ar}(\mathbf{x}_i)\mathbf{a} \\
&= \mathbf{a}^\top \mathbf{S}\mathbf{a} \\
&= \mathbf{a}^\top \mathbf{W}\mathbf{a} + \mathbf{a}^\top \mathbf{B}\mathbf{a}
\end{aligned}$$

which we have decomposed into the within-class variance of $\mathbf{z}$ and the between-class variance $\mathbf{z}$.

Fisher's criterion is to choose a vector, $\mathbf{a}$, to maximise the ratio of the **between-class** variance relative to the **within-class** variance of $\mathbf{z} = \mathbf{X}\mathbf{a}$, i.e., to solve

$$\max_{\mathbf{a}} \frac{\mathbf{a}^\top \mathbf{B}\mathbf{a}}{\mathbf{a}^\top \mathbf{W}\mathbf{a}}, \tag{8.7}$$

The idea is that this choice of $\mathbf{a}$ will make the classes most easily separable.

Solving the optimization problem

How do we solve the optimization problem (8.7) and find the optimal choice of $\mathbf{a}$?

Proposition 8.4. *A vector* $\mathbf{a}$ *that solves*

$$\max_{\mathbf{a}} \frac{\mathbf{a}^\top \mathbf{B}\mathbf{a}}{\mathbf{a}^\top \mathbf{W}\mathbf{a}}$$

is an eigenvector of $\mathbf{W}^{-1}\mathbf{B}$ *corresponding to the largest eigenvalue of* $\mathbf{W}^{-1}\mathbf{B}$.

Proof. Firstly, note that an equivalent optimization problem is

$$\begin{aligned}
&\text{Maximize } \mathbf{a}^\top \mathbf{B}\mathbf{a} \\
&\text{subject to } \mathbf{a}^\top \mathbf{W}\mathbf{a} = 1
\end{aligned}$$

as we can rescale $\mathbf{a}$ without changing the objective (8.7). This looks a lot like the optimization problems we saw in the chapters on PCA and CCA.

To solve this, note that if we write $\mathbf{b} = \mathbf{W}^{\frac{1}{2}}\mathbf{a}$ then the optimization problem becomes

$$\begin{aligned} &\text{Maximize } \mathbf{b}^\top \mathbf{W}^{-\frac{1}{2}} \mathbf{B} \mathbf{W}^{-\frac{1}{2}} \mathbf{b} \\ &\text{subject to } \mathbf{b}^\top \mathbf{b} = 1. \end{aligned}$$

Proposition 3.7 tells us that the maximum is obtained when $\mathbf{b} = \mathbf{v}_1$, where $\mathbf{v}_1$ is the eigenvector of $\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}$ corresponding to the largest eigenvalue of $\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}$, λ_1 say.

Converting back to $\mathbf{a}$ gives the solution to the original optimization problem (8.7) to be

$$\mathbf{a} = \mathbf{W}^{-\frac{1}{2}}\mathbf{v}_1$$

Note that this is an eigenvalue of $\mathbf{W}^{-1}\mathbf{B}$

$$\begin{aligned} \mathbf{W}^{-1}\mathbf{B}\mathbf{a} &= \mathbf{W}^{-1}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}\mathbf{v}_1 \\ &= \mathbf{W}^{-\frac{1}{2}}\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}\mathbf{v}_1 \\ &= \lambda_1 \mathbf{W}^{-\frac{1}{2}}\mathbf{v}_1 \\ &= \lambda_1 \mathbf{a} \end{aligned}$$

Finally, to complete the proof we should check that $\mathbf{W}^{-1}\mathbf{B}$ doesn't have any larger eigenvalues, but we can do this by showing that its eigenvalues are the same as the eigenvalues of $\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}$. This is left as an exercise (note we've already done one direction - all you need to do is show that if $\mathbf{W}^{-1}\mathbf{B}$ has eigenvalue λ then so does $\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}$). □

Fisher's discriminant rule

The function $L(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$ is called Fisher's linear discriminant function. Once $L(\mathbf{x})$ has been obtained, we allocate $\mathbf{x}$ to the population Π_k whose discriminant score $L(\hat{\boldsymbol{\mu}}_k)$ is closest to $L(\mathbf{x})$, that is, we use the discriminant rule

$$d^{Fisher}(\mathbf{x}) = \arg\min_k |L(\mathbf{x}) - L(\boldsymbol{\mu}_k)| = \arg\min_k |\mathbf{a}^\top \mathbf{x} - \mathbf{a}^\top \hat{\boldsymbol{\mu}}_k|.$$

This is of the same form as saw in the earlier sections. If we let $\delta_k(\mathbf{x}) = -|\mathbf{a}^\top(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)|$ then we see that $d^{Fisher}(\mathbf{x}) = \arg\max_k \delta_k(\mathbf{x})$ as before.

If there are only two populations (and suppose $L(\boldsymbol{\mu}_1) > L(\boldsymbol{\mu}_2)$), this is equivalent to classifying to population 1 if $L(\mathbf{x}) > t$ and to population 2 otherwise, where $t = \frac{1}{2}(L(\boldsymbol{\mu}_1) + L(\boldsymbol{\mu}_2))$.

This is visualized in Figure 8.5 for the iris data. The black points are the setosa data, and the green the viriginica. The diagonal black line is in the direction $\mathbf{a}$. Along that line we have plotted the projection of the data points onto the line $(\mathbf{x}^\top \mathbf{a})$, with the mean of each population and their projections marked with $+$. The red line is perpendicular to $\mathbf{a}$, and joins the midpoint of the two population means, $\mathbf{h} = \frac{\hat{\boldsymbol{\mu}}_s + \hat{\boldsymbol{\mu}}_v}{2}$, with the projection of that point onto $\mathbf{a}$. The red diamond marks the decision boundary for the projected points, i.e., if the point is to the left of this we classify as setosa, otherwise viriginica. It is half way between the projection of the two population means.

In Figure 8.5 $\mathbf{a}$ is chosen to be Fisher's optimal vector (the first eigenvector of $\mathbf{W}^{-1}\mathbf{B}$). This is the projection that is optimal for maximizing the ratio of the between-group to within-group variance, i.e., it optimally separates the two populations in the projection. In Figure 8.6 $\mathbf{a}$ is instead chosen to be the first principal component, i.e., the first eigenvector of the covariance matrix. This is the projection that maximizes the variance of the projected points (as done in PCA). Note that this is different to the LDA projection, and does not separate the two populations as cleanly as the LDA projection did.

Multiple projections

To summarize what we've found so far, we have seen that the vector $\mathbf{a}$ which maximizes

$$\frac{\mathbf{a}^\top \mathbf{B}\mathbf{a}}{\mathbf{a}^\top \mathbf{W}\mathbf{a}}$$

is the first eigenvector of $\mathbf{W}^{-1}\mathbf{B}$. As we did with PCA and CCA, we can continue and find a second, third, etc projection of the data. We're not going to go through the details of the derivation, but we can consider the projection matrix

$$\mathbf{A} = \begin{pmatrix} | & \dots & | \\ \mathbf{a}_1 & \dots & \mathbf{a}_r \\ | & \dots & | \end{pmatrix}$$

which has columns equal to the first r eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$.

Recall that

$$\mathbf{B} = \frac{1}{n}\sum_{j=1}^{g} n_j (\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})(\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}})^\top$$

is the variance of the g population means. The points $\hat{\boldsymbol{\mu}}_j - \bar{\mathbf{x}}$ lie in a $g-1$ dimensional subspace of $\mathbb{R}^p$ (they must sum to $\mathbf{0}$), and so $\mathbf{B}$ and also $\mathbf{W}^{-1}\mathbf{B}$ has rank at most $\min(g-1, p)$. Thus we can find at most $\min(g-1, p)$ projections of the data for maximizing the separation between the classes.

To classify points using r different projections, we use the vector discriminant function

$$\mathbf{L}(\mathbf{x}) = \mathbf{A}^\top \mathbf{x}$$

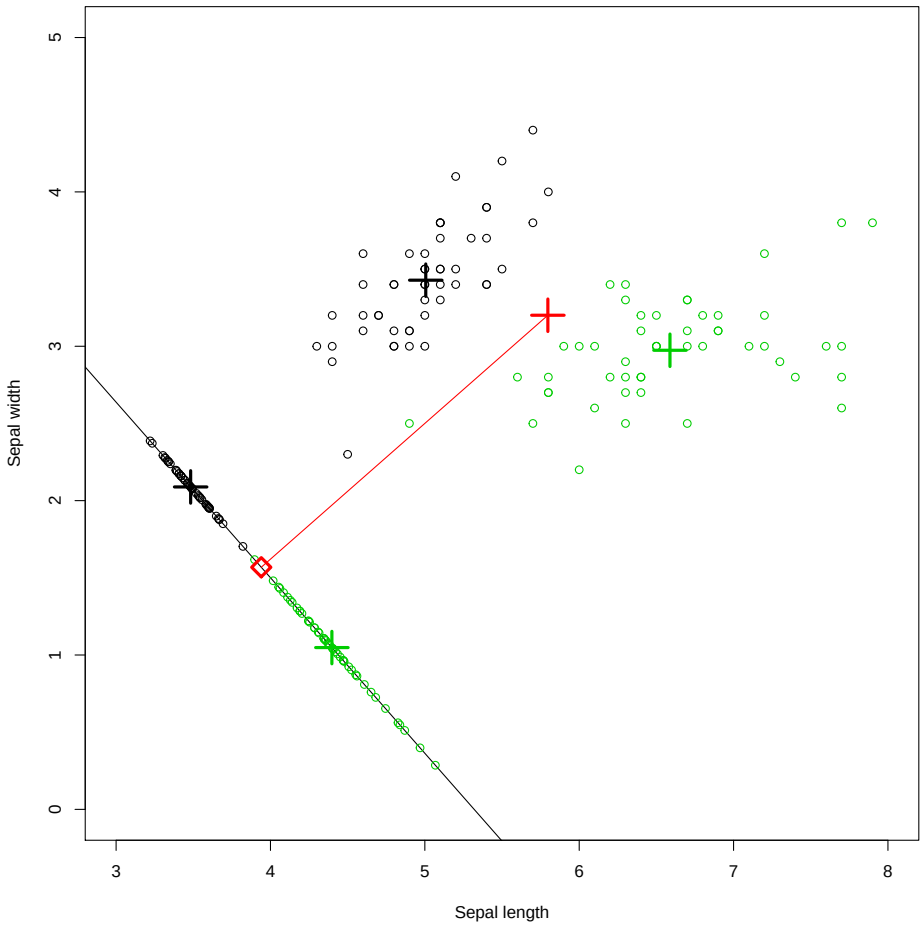


Figure 8.5: Visualization of Fishers discriminant analysis.

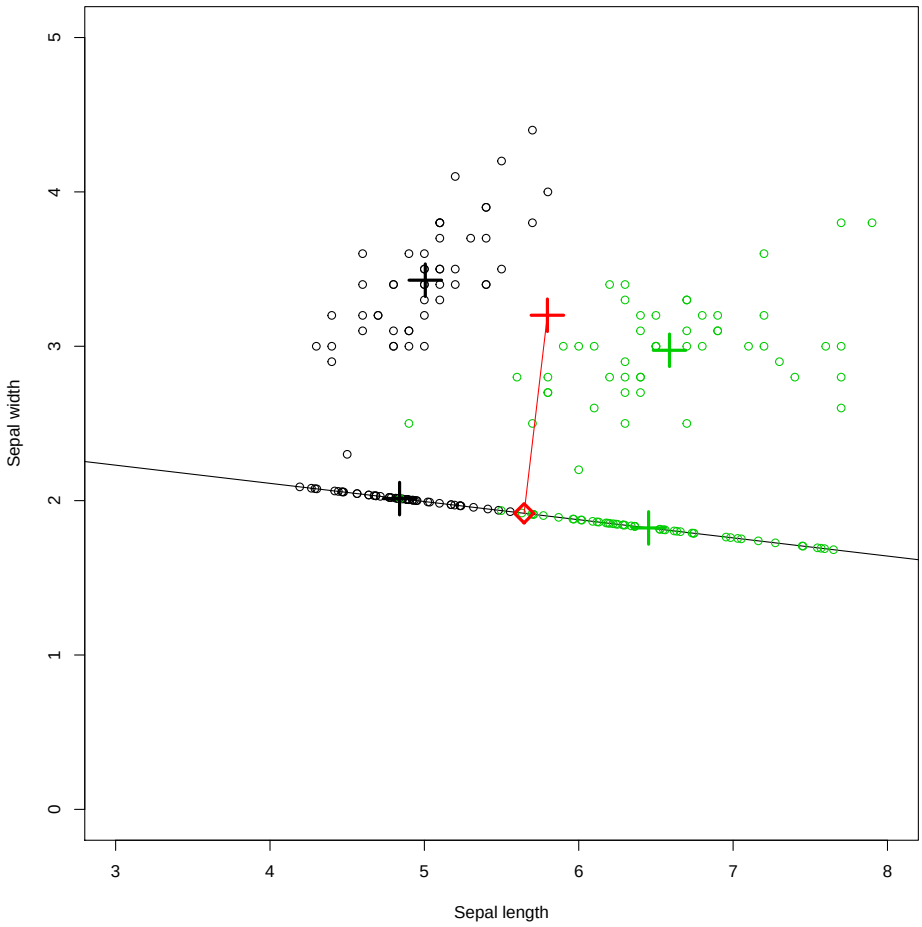


Figure 8.6: Projection onto the first principal component.

and use the discriminant rule

$$d^{Fisher}(\mathbf{x}) = \arg\min_k ||\mathbf{L}(\mathbf{x}) - \mathbf{L}(\boldsymbol{\mu}_k)||_2,$$

i.e., we compute the r dimensional projection of the data and then find the which (projected) population mean it is closest to.

Note the dimension reduction here. We have gone from an observation $\mathbf{x} \in \mathcal{R}^p$ to a projected point $\mathbf{A}^\top \mathbf{x} \in \mathcal{F}^t$. We are free to choose r, which can result in useful ways of visualizing the data.

8.3.1 Iris example continued

Let's continue the iris example we had before, using the full dataset of 150 observations on 3 species, with 4 measurements on each flower.

```
library(vcvComp)
B=cov.B(iris[,1:4], iris[,5])
W=cov.W(iris[,1:4], iris[,5])
iris.eig <- eigen(solve(W)%*% B)
iris.eig$values
```

```
## [1] 4.732214e+01 4.195248e-01 1.426781e-14 6.229462e-16
```

We can see that $\mathbf{W}^{-1}\mathbf{B}$ only has two positive eigenvalues, and so is rank 2. We expected this, as we said the rank of $\mathbf{W}^{-1}\mathbf{B}$ must be less than $\min(g-1, p) = \min(2, 4) = 2$. We can plot these projections and look at the separation achieved.

```
V <- iris.eig$vectors[,1:2]
Z <- as.matrix(iris[,1:4])%*% V
ggplot2::qplot(Z[,1], Z[,2], colour=iris$Species, xlab='LD1',ylab='LD2')
```

We can see that the first projected coordinate is doing nearly all of the useful work in separating the three populations. We can see this in the eigenvalues (c.f. the scree plots used in PCA) which can be used to understand the importance of each projected coordinate in separating the populations (this is the `Proportion of trace` reported in the `lda` output).

```
iris.eig$values[1:2]/sum(iris.eig$values)
```

```
## [1] 0.991212605 0.008787395
```

R will automatically plot this projection for us:

```
par(pty="s")
iris.lda <- lda(Species ~., iris)
plot(iris.lda,col=as.integer(iris$Species)+1, abbrev=1)
```

Although this looks different to Figure 8.7, the projection is essentially the same it has just flipped the y axis.

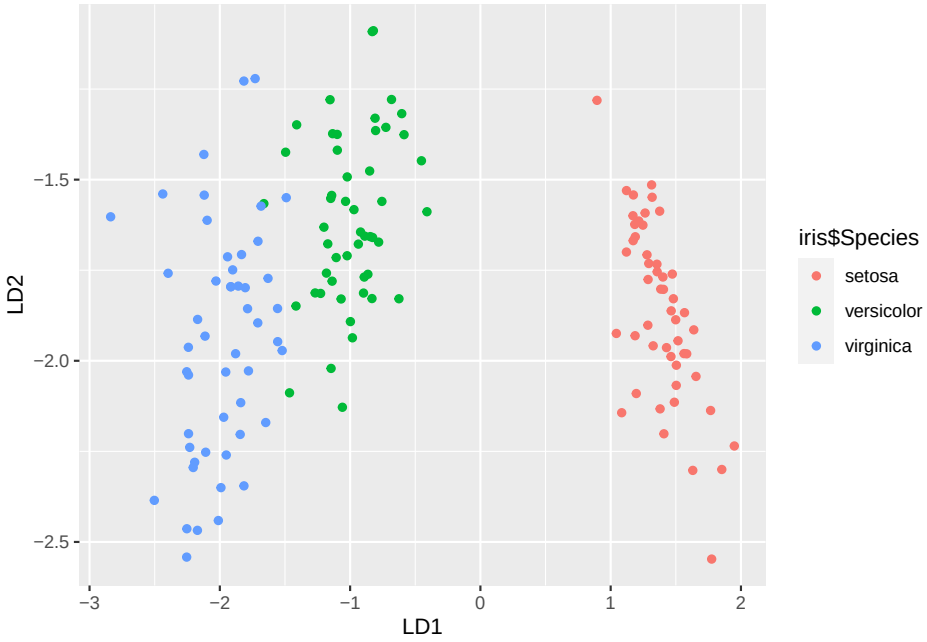


Figure 8.7: 2-dimensional LDA projection of the iris data

Using the option `dimen = 1` will plot histograms of the first LDA projection for each group, which all we need in this example to separate the populations.

```
plot(lda.iris, dimen = 1, type = "b")
```

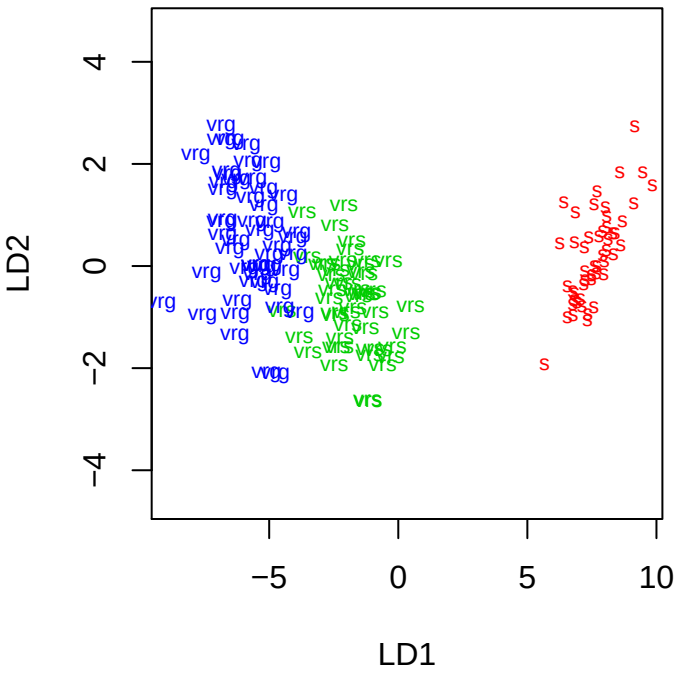
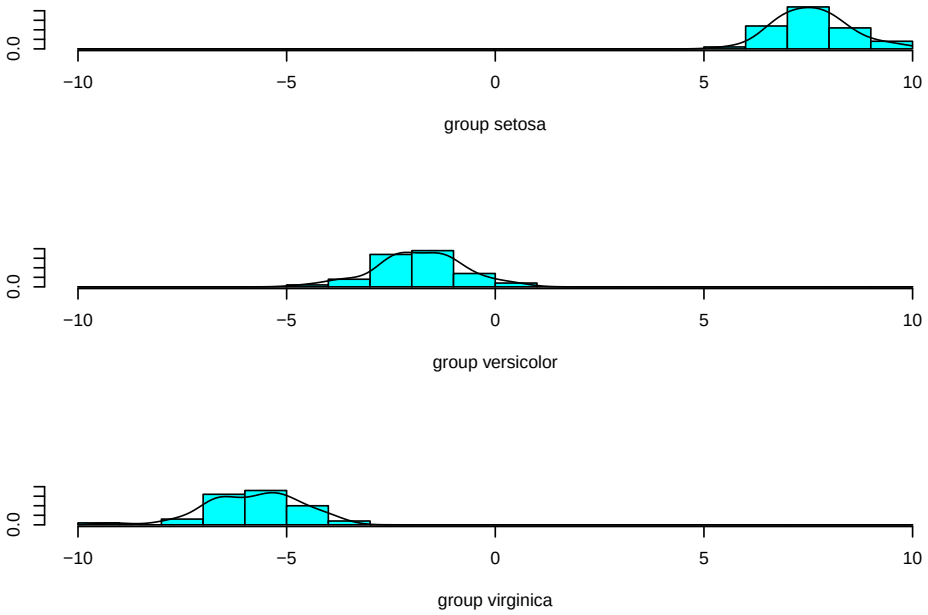


Figure 8.8: 2-dimensional LDA projection of the iris data using the built-in R plotting function



8.3.1.0.1 Comparison with PCA

We can compare the LDA projection to the 2-dimensional projection found by PCA:

```
iris.pca = prcomp(iris[,1:4])

iris$PC1=iris.pca$x[,1]
iris$PC2=iris.pca$x[,2]
ggplot2::qplot(PC1, PC2, colour=Species, data=iris)
```

PCA hasn't separated the 3 populations as cleanly as LDA did, but that is not a surprise as that is not its aim (it aim is to maximize the variance). PCA doesn't use the population labels, and so there is no reason why it should separate the populations. But this does illustrate the importance of choosing the right multivariate statistical methods.

8.3.2 Links between methods

When $g = 2$, Fisher's rule and the sample ML rule with $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$ turn out to be the same. Note that in the sample ML rule we assumed that the two groups are from $N_p(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$ populations, but Fisher's rule makes no such assumption.

Proposition 8.5. *If $g = 2$ then Fisher's rule and the sample ML rule described in Section 8.1.2 are equivalent.*

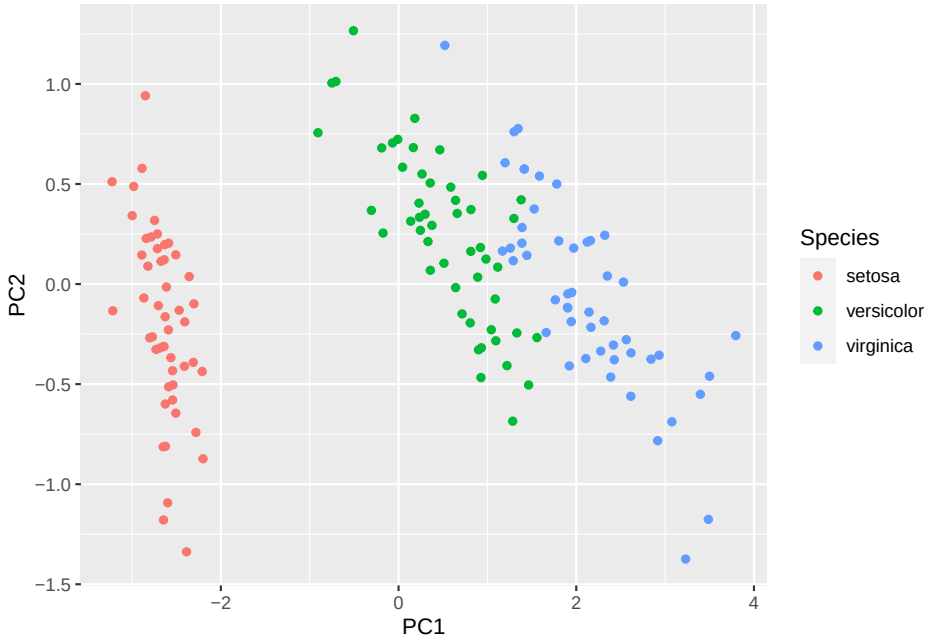


Figure 8.9: 2-dimensional PCA projection of the iris data

Proof. Beyond the scope of the module. □

For $g > 2$, the sample ML rule and Fisher's linear rule will not, in general, be the same. Fisher's rule is linear when $g > 2$ and is easier to implement than ML rules when there are several populations. It is often reasonable to use Fisher's rule for non-normal populations. In particular, Fisher's rule requires fewer assumptions than ML rules. However, the ML rule is 'optimal' in some sense when its assumptions are valid.

There is also a link between LDA and linear regression when there are only two groups. If we let

$$y_i = \begin{cases} 1 & \text{if } i \text{ is from } \Pi_1 \\ -1 & \text{otherwise} \end{cases}$$

then if we fit the model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

we find $\boldsymbol{\beta}$ is in the same direction as the first eigenvector of $\mathbf{W}^{-1}\mathbf{B}$.

```
library(dplyr)
iris3 <- iris %>% filter(Species != "versicolor") %>%
```

```
  dplyr::select(Sepal.Length, Sepal.Width, Species)
iris3$pm1 <- as.integer(iris3$Species)-2
# convert to +1 and -1
iris3.lm <- lm(pm1~Sepal.Width+Sepal.Length, iris3)
coef(iris3.lm)[2]/coef(iris3.lm)[3]
```

```
## Sepal.Width
##   -1.137257
```

```
iris.lda <- lda(Species~Sepal.Width+Sepal.Length, iris3)
coef(iris.lda)[1]/coef(iris.lda)[2]
```

```
## [1] -1.137257
```

8.4 Computer tasks

8.4.0.0.1 Task 1

Let's look again at the crabs dataset.

```
library(MASS)
data(crabs)
head(crabs)
```

i. Use the `lda` command to build a classifier to predict the `sex` of the crabs from the five numerical measurements (don't use the index!).

ii. Test the predictive accuracy of your classifier by splitting the data into a training set and a test set. Report the predictive accuracy and find the confusion matrix.

iii. Plot the histograms of the 1d projections of the data. Note that there can only be single projected variable here as there are just $g = 2$ groups.

iv. Use your classifier to predict the sex of a crab that has BD=14, FL=15.5, RW=13.3, CL=31, and CW=36. What probability does the classifier give for this crab being male?

v. Create a new variable that indicates the species and the sex of the crab. With levels BM, BF, OM, OF. This can be done as follows:

```
crabs$spsex <- factor(paste(crabs$sp, crabs$sex, sep=""))
```

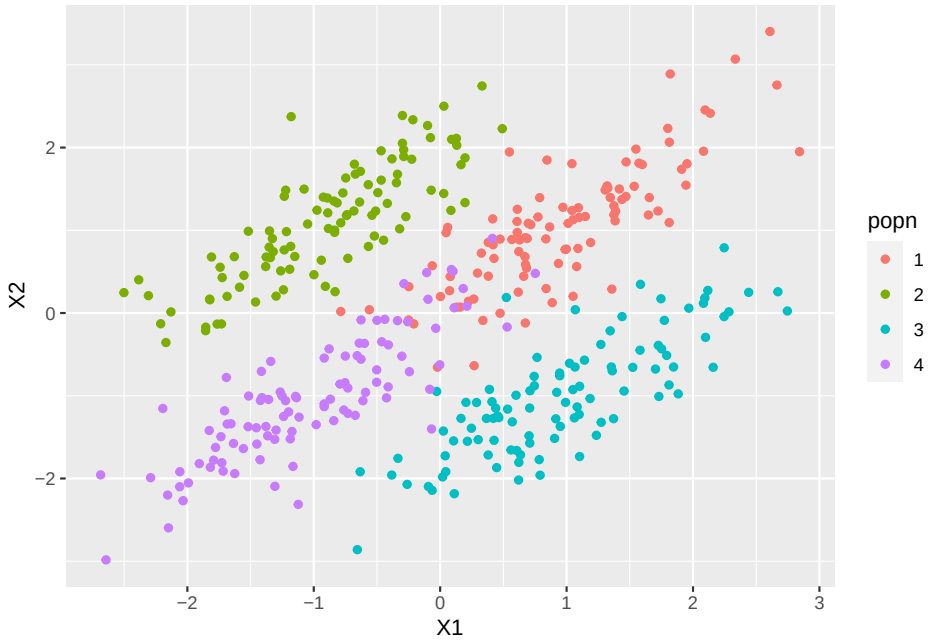
Build a classifier to predict the species and sex of the crabs. Test its predictive accuracy, and provide some plots showing its effectiveness etc.

8.4.0.0.2 Task 2

In this question we will generate some data ourselves, and then see how successful LDA is at separating the populations.

i. Generate 4 populations in 2d using the MVN distribution as follows:

```
mu1 <- c(1,1)
mu2 <- c(-1,1)
mu3 <- c(1,-1)
mu4 <- c(-1,-1)
Sigma <- matrix(c(0.5,0.4,0.4,0.5), nr=2)
library(mvtnorm)
S1 <- rmvnorm(100, mu1, Sigma)
S2 <- rmvnorm(100, mu2, Sigma)
S3 <- rmvnorm(100, mu3, Sigma)
S4 <- rmvnorm(100, mu4, Sigma)
X=rbind(S1,S2,S3,S4)
dat <- data.frame(popn=c(rep("1",100),rep("2",100),rep("3",100),rep("4",100)), X1=X[,1]
library(ggplot2)
qplot(x=X1,y=X2, colour=popn, data=dat)
```



ii. Use LDA to train a classifier. Plot the 2d projection found, and use the `partimat` command from the `klaR` package to visualise the discriminant regions.

iii. Experiment with different population means, different number of populations, and different covariance functions. What makes populations easy/hard to separate?

8.4.0.0.3 Task 3

With a bit of work, it is possible to get a prediction accuracy of over 80% for the MNIST data using linear discriminant analysis.

i. Create a training set of 1000 images, and try using the `lda` command to fit a linear classifier. Did it work?

```
load('mnist.rda')
X<- as.matrix(mnist$train$x[1:10000,])
y<-mnist$train$y[1:10000]
library(MASS)
lda(X,y)
```

ii. One way to fix problems such as colinearity, or in this case, zero variance, is to first use PCA on the data to rotate to a set of variables with maximal variance. Do PCA on your training data using just the Xs (the pixel intensities) and select the $p = 100$ most variable PC scores. This should leave you with a 1000×100 matrix.

iii. Do linear discriminant on the 100 PC variables you derived in the previous part. Plot the LDA projections of the data: Try plotting both the first 2 projected variables (coloured by the digit they represent), and the first 3 projected variables. Does using 3 dimensions help in separating the different populations?

iv. Find the predictive accuracy of your classifier using the MNIST test data. Note that you will first need to project this data onto the p leading principal components. Find the confusion matrix and comment upon it.

```
Xtest <- as.matrix(mnist$test$x)
Ytest <- as.matrix(mnist$test$y)
```

v. Does the predictive accuracy change if instead of using the first $p = 100$ principal component scores you use fewer or more? You can also try using larger training sets. The MNIST training data consists of $60,000$ images - depending upon your computer you may be able to repeat the analysis with all $60,000$ images. If so, how does this affect the prediction accuracy?

8.5 Exercises

1. Prove that the Bayes classifier can be written as given in Proposition 8.3.
2. Prove that $\mathbf{W}^{-1}\mathbf{B}$ and $\mathbf{W}^{-\frac{1}{2}}\mathbf{B}\mathbf{W}^{-\frac{1}{2}}$ have the same eigenvalues and find an expression relating their eigenvectors.
3. Prove Proposition 8.1. Note that region $\mathcal{R}$ is convex and connected if for $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}$ we have

$$\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathcal{R} \text{ for all } \lambda \in [0,1].$$

4. Consider $g = 3$ bivariate normal populations with the same covariance matrix, given by

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}.$$

Determine the three maximum likelihood discriminant rules and, hence, sketch the ML discriminant regions. Determine the point at which the three boundary lines meet.

5. The`palmerpenguins` R package contains size measurements of adult foraging penguins near Palmer Station, Antarctica. We will begin by looking at a subset of this data, looking at just the *Adelie* ($n_1 = 151$) and *Chinstrap* ($n_2 = 68$) penguins, and just the *bill length* and *bill depth* measurements. The sample mean and covariance for each group are

$$\boldsymbol{\mu}_A = \begin{pmatrix} 38.8 \\ 18.3 \end{pmatrix} \qquad \boldsymbol{\Sigma}_A = \begin{pmatrix} 7.09 & 1.27 \\ 1.27 & 1.48 \end{pmatrix}$$
$$\boldsymbol{\mu}_C = \begin{pmatrix} 48.8 \\ 18.4 \end{pmatrix} \qquad \boldsymbol{\Sigma}_C = \begin{pmatrix} 11.2 & 2.48 \\ 2.48 & 1.29 \end{pmatrix}$$

i. Find the ML discriminant rule for allocating a new penguin $\mathbf{z} = (z_1, z_2)^T$ as either *Adelie* or *Chinstrap* assuming that the population covariance matrices for the two groups are equal. In particular, determine the sample ML rule for allocating the new observation $\mathbf{z}$. Find the equation of the straight line which separates the two allocation regions and plot the two regions graphically.

ii. Find the Bayes discriminant rule estimating the prior probabilities using the observed frequencies n_1 and n_2. How does this differ to the ML rule?

iii. In addition to the *Adelie* and *Chinstrap* penguins, there are also $n_3 = 123$ measurements on *Gentoo* penguins with sample mean and covariance

$$\boldsymbol{\mu}_G = \begin{pmatrix} 47.5 \\ 15 \end{pmatrix} \qquad \boldsymbol{\Sigma}_G = \begin{pmatrix} 9.5 & 1.95 \\ 1.95 & 0.963 \end{pmatrix}$$

The within-group covariance matrix is

$$\frac{1}{342}(151\mathbf{S}_A + 68\mathbf{S}_C + 123\mathbf{S}_G) = \begin{pmatrix} 9.25 & 1.9 \\ 1.9 & 1.24 \end{pmatrix}.$$

The eigendecomposition of $\mathbf{W}^{-1}\mathbf{B}$ is found in R to be

```
WiB.eig
```

```
## eigen() decomposition
## $values
```

```
## [1] 9.847607 1.269243
##
## $vectors
##            [,1]      [,2]
## [1,]  0.3470103 0.3350582
## [2,] -0.9378613 0.9421974
```

Calculate Fisher's linear discriminant function using just a single LDA variable.

iv. What would Fisher's discriminant rule be if you use both LDA variables?

Chapter 9

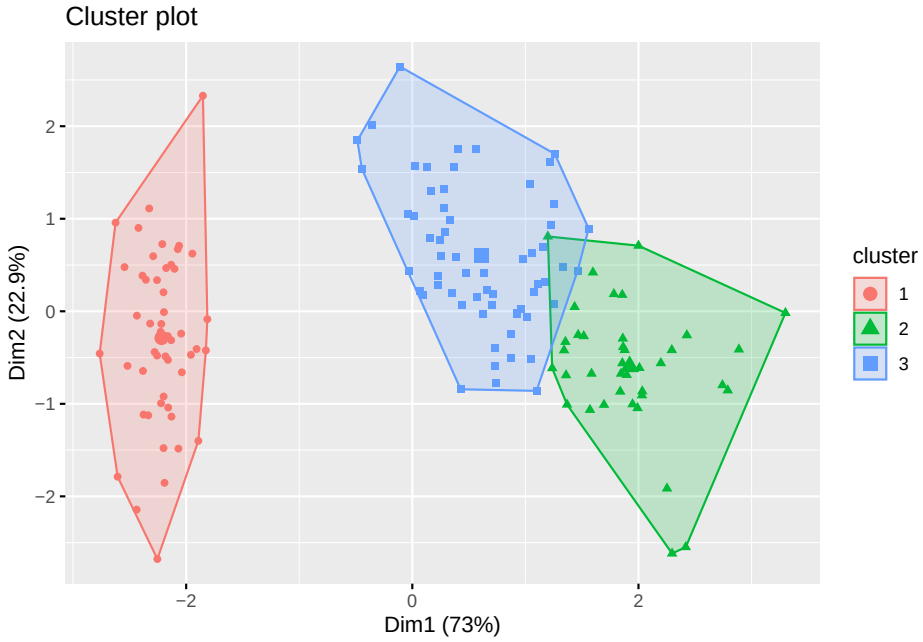
Cluster Analysis

Discriminant analysis, covered in Chapter 8, is a **supervised** learning method: in order to train the classifier we had access to both the input $\mathbf{x}$ and the label y for that case (what group it belonged to). This chapter focusses on cluster analysis, which is an **unsupervised** learning method: we have to train the model without knowledge of the true cluster labels. In many problems there may not be anything we could describe as a 'true' cluster label - in which case we are using cluster analysis as a way finding descriptive statistics about the data.

The aim is to group cases into 'clusters' such that cases within each cluster are more closely related to each other than to cases in other clusters. Some methods do this using the attributes/measurements $\mathbf{x}$ for each case. These methods include k-means clustering and model-based clustering. In contrast, other methods such as hierarchical clusteringmethods, use the proximities or distances between cases (as in MDS).

The videos for this chapter are available at the following links:

- Brief introduction to clustering
- K-means clustering
- K-means example
- Model-based clustering
- Hierarchical clustering
- Hierarchical clustering example



9.1 K-means clustering

Suppose we have a sample of n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$. Consider the situation where $\mathbf{x}_i$ comes from one of K sub-populations (I used g previously, but this method is known as 'K'-means so we'll use K instead of g here). We'll initially assume K is known, but will discuss how to choose K later.

The key point is that we do not know which sub-population each $\mathbf{x}_i$ comes from, i.e., we do not know how to allocate the observations to sub-populations. **The goal of cluster analysis** is to allocate each case, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to one of K **clusters**, $\mathcal{C}_1, \ldots, \mathcal{C}_k$, in such a way that observation vectors within a cluster tend to be more similar to each other, in some sense, than to observations in different clusters.

Each data point, $\mathbf{x}_i$, will be allocated to precisely one cluster:

$$\mathbf{x}_i \in \mathcal{C}_j$$

so that the clusters partition the n cases:

$$\bigcup_{j=1}^{k} \mathcal{C}_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \qquad \text{and} \qquad \mathcal{C}_j \cap \mathcal{C}_{j'} = \emptyset, \qquad j \neq j'.$$

It will be convenient to introduce an alternative, but equivalent, way to describe the allocation of cases to clusters. We will do this with an **encoder**, $\delta(i)$, for $i = 1, \ldots, n$, with:

$$\delta(i) = j \iff \mathbf{x}_i \in \mathcal{C}_j. \tag{9.1}$$

We write $\boldsymbol{\delta} = (\delta(1), \ldots, \delta(n))^\top$ for the vector of n different cluster indicators.

9.1.1 Estimating $\boldsymbol{\delta}$

We think of the encoder $\delta(i)$ as a parameter that we need to estimate: $\boldsymbol{\delta} \in \mathbb{N}^n$. We estimate it by picking a loss function, and then seeking to minimize that loss. A natural choice for the loss function is to use the within-cluster scatter that we saw previously:

$$W(\delta) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i:\delta(i)=k} \sum_{i':\delta(i')=k} d(\mathbf{x}_i, \mathbf{x}_{i'})$$

for some distance function d. As in Chapter 8, we can split the total point scatter into within-cluster and between-cluster scatter:

$$T = W(\boldsymbol{\delta}) + B(\boldsymbol{\delta}) \tag{9.2}$$

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} d(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i:\delta(i)=k} \left(\sum_{i':\delta(i')=k} d(\mathbf{x}_i, \mathbf{x}_{i'}) + \sum_{i':\delta(i')\neq k} d(\mathbf{x}_i, \mathbf{x}_{i'}) \right) \tag{9.3}$$

and so minimizing within-cluster scatter is equivalent to maximizing between-cluster scatter.

In general, solving the optimization problem

$$\hat{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\delta}} W(\boldsymbol{\delta})$$

is impossible in most cases, as the number of possible allocations of cases to clusters explodes combinatorially. So instead of searching through all possible choices of $\boldsymbol{\delta}$ we take an iterative greedy descent approach. These work by

1. Picking an initial partition $\boldsymbol{\delta}$.
2. At each step, the cluster assignments are changed to reduce the loss function $W(\boldsymbol{\delta})$

9.1.2 K-means

K-means clustering is the most commonly used iterative descent clustering method. It uses the squared Eucliden distances between points

$$d(\mathbf{x}_i, \mathbf{x}_{i'}) = ||\mathbf{x}_i - \mathbf{x}_{i'}||_2^2.$$

In this case, the within-cluster scatter reduces to

$$W(\delta) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i:\delta(i)=k}\sum_{i':\delta(i')=k}||\mathbf{x}_i - \mathbf{x}_{i'}||_2^2 \tag{9.4}$$

$$= \sum_{k=1}^{K} n_k \sum_{i:\delta(i)=k}||\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k||_2^2 \tag{9.5}$$

$$\tag{9.6}$$

where $n_k = \sum_{i=1}^{n}\mathbb{I}_{\delta(i)=k}$ is the number of points assigned to cluster k, and

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i:\delta(i)=k}\mathbf{x}_i$$

is the mean vector of the points assigned to cluster k (you'll be asked to prove this in the Exercises).

Thus we can see that k-means aims to minimize the sum of the square distance from each point to its cluster mean.

The greedy iterative approach used in K-means clustering is thus as follows:

1. For a given cluster assignment $\boldsymbol{\delta}$, find the mean of each cluster $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K$.
2. Given a set of cluster means $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K$, allocate each point to the closest cluster mean, i.e., set
$$\delta(i) = \arg\min_k ||\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k||_2^2$$
3. Iterate steps 1. and 2. until convergence.

Note that the result of these steps might be a sub-optimal local minimum. Thus, we usually run K-means several times with different random initial choices for $\boldsymbol{\delta}$, and then choose the best solution.

9.1.3 Example: Iris data

Consider the *iris* data again. Suppose we are given the petal and sepal length and widths, but not told which species each iris belongs to. I.e., suppose we are given the data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

Our goal is to find clusters within the data. These are groups of irises that are more similar to each other than to those in other groups (clusters). These clusters may correspond to the Species label (which we aren't given), or they may not. The goal of cluster analysis is not to predict the species, but simply to group the data into similar clusters.

Let's look at finding 3 clusters. We can do this using the `kmeans` command in R.

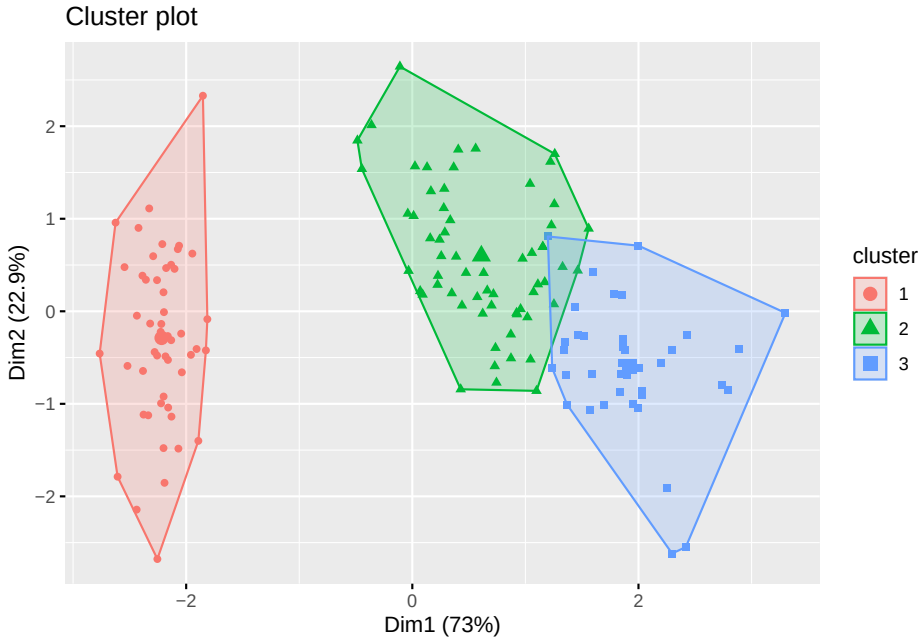
```
iris2 <- iris[,1:4]
# nstart gives the number of random initialisations to try
set.seed(123)
(iris.k <- kmeans(iris2, centers = 3, nstart=25))
```

```
## K-means clustering with 3 clusters of sizes 50, 62, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     5.901613    2.748387     4.393548    1.433871
## 3     6.850000    3.073684     5.742105    2.071053
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [149] 3 2
##
## Within cluster sum of squares by cluster:
## [1] 15.15100 39.82097 23.87947
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

From this output we can read off the final cluster means, and the encoder $\boldsymbol{\delta}$. Also given is the final within-cluster sum of squares for each cluster.

We can visualise the output of K-means using the `fviz_cluster` command from the `factoextra` package. This first projects the points into two dimensions using PCA, and then shows the classification in 2d, and so some caution is needed in interpreting these plots.

```
library(factoextra)
fviz_cluster(iris.k, data = iris2,
             geom = "point")
```



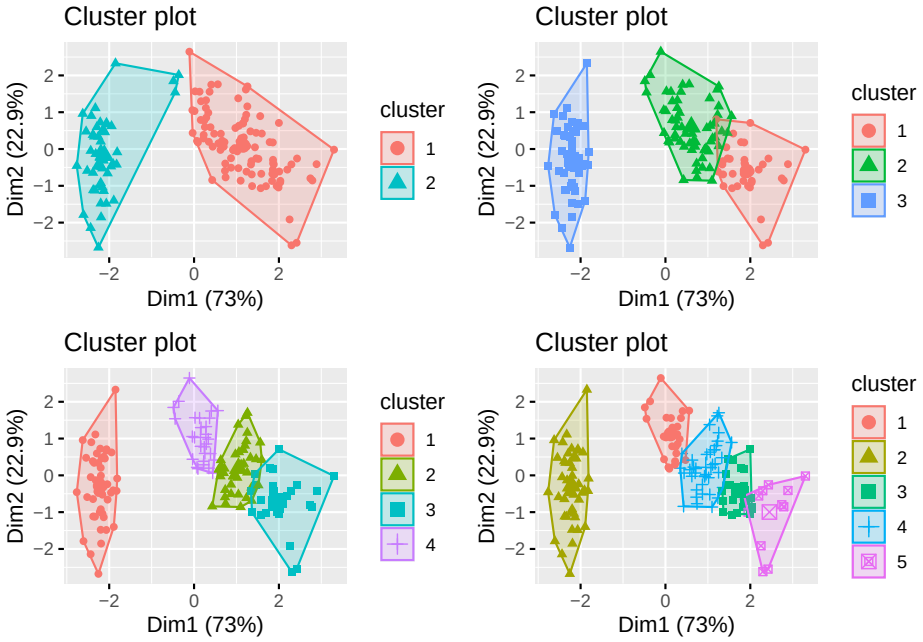
Finally, in this case we know that there really are three clusters in the data (the three species). We can compare the clusters found using K-means with the species label to see if they are similar. The easiest way to do this is with the `table` command.

```
table(iris$Species, iris.k$cluster)
```

```
##
##                1  2  3
##    setosa     50  0  0
##    versicolor  0 48  2
##    virginica   0 14 36
```

9.1.4 Choosing K

In the iris data, we know there are 3 distinct species. But suppose we didn't know this. What happens if we try other values for K?

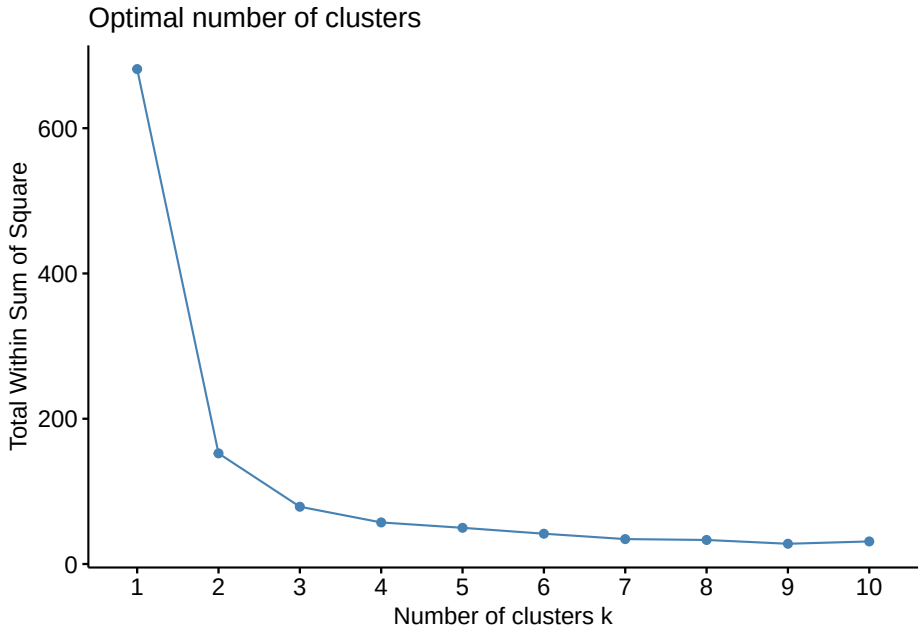


How do we choose between these different values of K? In general, if we choose K too large, then each individual case will begin to represent a cluster. Conversely, if K is too small then we will find data points are incorrectly clustered together.

The best approach for choosing K is to use domain knowledge when it is available. In the case where there is no prior knowledge of how many clusters we expect to find, we can use a data-driven approach for choosing K. A common approach data-driven approach is known as the **elbow method**. This approach looks at the within-cluster sum of squares $W(\delta)$ as the number of clusters changes, i.e., we compute $W_1, W_2, \ldots, W_{K_{max}}$. W will decrease as the number of clusters increases, but we can look for the point where the decrease slows down. The intuition is that if we've used K clusters when really the data consist of $K+1$ clusters, $W_K - W_{K+1}$ will be large. In constrast, if there are really K clusters, and we use $K^* > K$ then at least one of the estimated clusters must partition one of the natural groups, and so the change in W will be less. As you can see, this is not an exact science!

For the iris data, we can create an elbow plot using the `fviz_nbclust` command from the `factoextra` package.

```
fviz_nbclust(iris2, kmeans, method = "wss")
```



In this case, I would probably decide there most likely three natural clusters in the data, as there is a reasonable decrease in W when moving from 2 to 3 clusters, but moving to 3 clusters only yields a minor improvement. Note here the slight increase in W in moving from 9 to 10 clusters. This is due to only using a greed search, rather than an exhaustive one (we know the best 10-group cluster must be better than the betst 9-group cluster, we just have found it).

9.2 Model-based clustering

Model-based clustering is similar to K-means clustering, in that we want to allocate each case to a cluster. The difference is that we will now assume a probability distribution for the observations within each cluster.

Consider the situation where $\mathbf{x}_i$ comes from one of K sub-populations, where the jth sub-population has probability density function $f(\mathbf{x}; \boldsymbol{\theta}_j)$, $j = 1, \ldots, K$. Here $\boldsymbol{\theta}_j$ are unknown parameters describing each sub-population.

The most common choice for the density f is to assume it is a multivariate normal distribution. The parameters $\boldsymbol{\theta}_j$ would then represent the mean and variance of the MVN density for each cluster.

9.2.1 Maximum-likelihood estimation

We want to estimate the optimal allocation of each case to a cluster, i.e., estimate $\boldsymbol{\delta}$, as well as the (usually) unknown parameter vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$. We can do this by maximum likelihood estimation.

The likelihood for $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ and the allocation $\boldsymbol{\delta}$ may be written as

$$\begin{aligned} L(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; \boldsymbol{\delta}) &= \left\{ \prod_{\mathbf{x} \in \mathcal{C}_1} f(\mathbf{x}; \boldsymbol{\theta}_1) \right\} \cdots \left\{ \prod_{\mathbf{x} \in \mathcal{C}_K} f(\mathbf{x}; \boldsymbol{\theta}_K) \right\} \\ &= \left\{ \prod_{i:\delta(i)=1} f(\mathbf{x}_i; \boldsymbol{\theta}_1) \right\} \cdots \left\{ \prod_{i:\delta(i)=K} f(\mathbf{x}_i; \boldsymbol{\theta}_K) \right\} \\ &= \prod_{i=1}^{n} f(\mathbf{x}_i; \boldsymbol{\theta}_{\delta(i)}) \end{aligned}$$

Let $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K$ and $\hat{\boldsymbol{\delta}}$ denote the maximum likelihood estimators of $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ and the unknown allocation $\boldsymbol{\delta}$. Also, let $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_K$ denote the maximum likelihood clusters, which are determined by $\hat{\boldsymbol{\delta}}$ via Equation (9.1).

Then we have the following result.

Proposition 9.1. *Observations are classified to the cluster for which they have the largest likelihood:*

$$\hat{\delta}(i) = \arg\max_k f(\mathbf{x}_i; \hat{\boldsymbol{\theta}}_k).$$

Proof. Suppose observation i, $\mathbf{x}_i$, gets classified to cluster $\hat{\mathcal{C}}_j$ according to the maximum likelihood estimator, i.e.,

$$\hat{\delta}(i) = j$$

Then consider the encoder $\boldsymbol{\delta}$ that is the same as $\hat{\boldsymbol{\delta}}$, but which moves $\mathbf{x}_i$ from $\hat{\mathcal{C}}_j$ to $\hat{\mathcal{C}}_k$.

$$\delta(i') = \begin{cases} \hat{\delta}(i') & \text{for } i' \neq i \\ k & \text{for } i' = i \end{cases}$$

Then the likelihood of $\boldsymbol{\delta}$ is

$$L(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K; \boldsymbol{\delta}) = L(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K; \hat{\boldsymbol{\delta}}) f(\mathbf{x}_i; \hat{\boldsymbol{\theta}}_k) / f(\mathbf{x}_i; \hat{\boldsymbol{\theta}}_j).$$

But by definition

$$L(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K; \boldsymbol{\delta}) \leq L(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K; \hat{\boldsymbol{\delta}}),$$

as $\hat{\boldsymbol{\delta}}$ maximizes the likelihood, and so we conclude that

$$f(\mathbf{x}; \hat{\boldsymbol{\theta}}_k) / f(\mathbf{x}; \hat{\boldsymbol{\theta}}_j) \leq 1$$

for all k and thus Proposition 9.1 holds. □

Note the similarity to the sample ML discriminant rule considered in Section 8.1: in each case we classify $\mathbf{x}$ to the cluster for which the likelihood $f(\mathbf{x}; \hat{\boldsymbol{\theta}}_k)$ is greatest. The difference with LDA is that we are not given the cluster labels for the training data, and so we have to estimate $\boldsymbol{\delta}$ as well as the different parameter vectors, $\boldsymbol{\theta}_k$, for each cluster.

The computation for model-based clustering is beyond the scope of the module, but is covered in the Computational Statistics module (the section on the EM algorithm).

9.2.2 Multivariate Gaussian clusters

We now consider the case where the sub-populations are multivariate Gaussian, i.e. $f(\mathbf{x}; \boldsymbol{\theta}_j)$ is the density of $N_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ for $j = 1, \ldots, K$. Here, $\boldsymbol{\theta}_j$ consists of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ for each $j = 1, \ldots, K$.

In the general case, when the mean vector and covariance matrix are different for each sub-population, we know how to maximise the likelihood when the allocation $\boldsymbol{\delta}$ is given (see Section 8.1.2). Conversely, given estimated parameters $\hat{\boldsymbol{\theta}}_k$, we can estimate the encoder $\boldsymbol{\delta}$ using Proposition 9.1. As for K-means, the procedure for doing model based clustering iterates between the steps

1. Estimate $\boldsymbol{\theta}_k$ given the current partition $\boldsymbol{\delta}$.
2. Estimate $\boldsymbol{\delta}$ given parameter values $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_k$.

until convergence.

9.2.2.1 Link to K-means clustering

The case where $\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \sigma^2 \mathbf{I}_p$, i.e. the common covariance matrix is a scalar multiple of the $p \times p$ identity matrix, turns out to be equivalent to K-means clustering. To see this, note that the maximum likelihood allocation $\hat{\boldsymbol{\delta}}$ given estimated cluster means $\hat{\boldsymbol{\mu}}_k$, is obtained by the $\boldsymbol{\delta}$ which maximizes

$$\prod_{i=1}^{n} \exp(-\frac{1}{2\sigma} ||\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{\delta(i)}||_2^2)$$

which is equivalent to minimizing

$$\sum_{j=1}^{K} \sum_{\mathbf{x} \in \mathcal{C}_j} ||\mathbf{x} - \hat{\boldsymbol{\mu}}_j||^2.$$

which in turn is equivalent to minimizing $W(\boldsymbol{\delta})$.

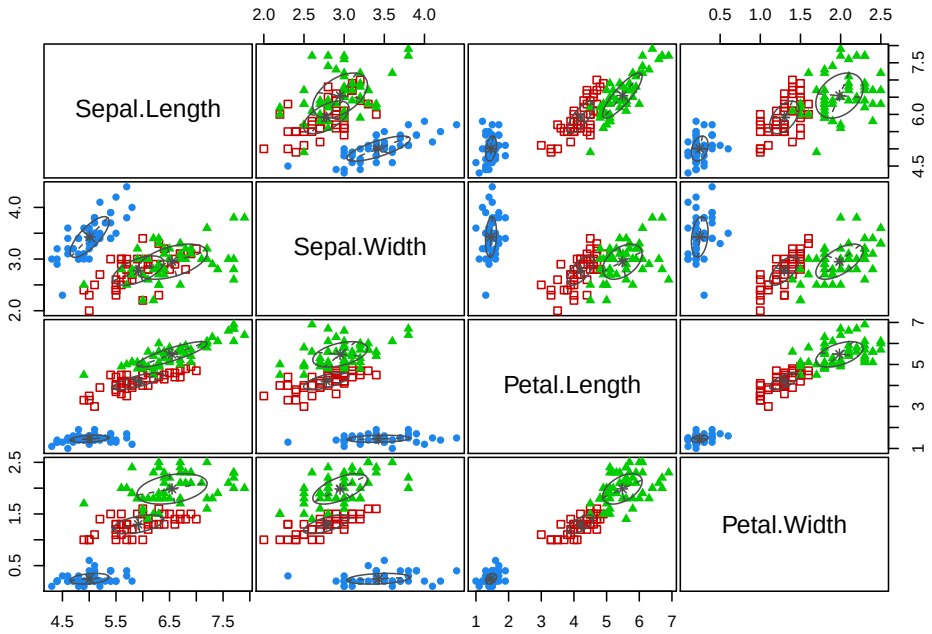
9.2.3 Example: Iris

The `mclust` library can be used to perform model-based clustering with Gaussian clusters. We just have to specify the number of required clusters.

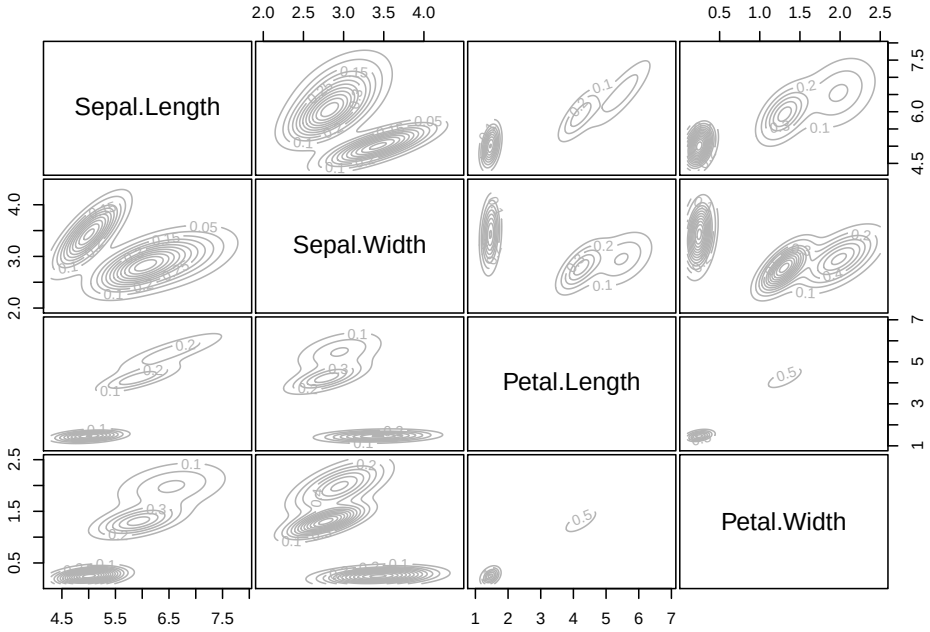
```
library(mclust)
iris.m <- Mclust(iris2,G=3)
```

Pairs plots of the classification of each point can easily be obtained, as can the estimated probability density of each cluster.

```
plot(iris.m, what = c("classification"))
```




```
plot(iris.m, what = c("density"))
```



How to choose K in model-based clustering is beyond the scope of the module.

9.3 Hierarchical clustering methods

Hierarchical clustering methods work by creating a hierarchy of clusters, in which clusters at each level of the heirarchy are formed by merging or splitting clusters from a neighbouring level of the hierarchy. A hierarchical clustering method is usually of one of two types:

1. **agglomerative** clustering methods start with the finest partition (one observation per cluster) and progressively combine clusters.
2. **divisive/splitting** clustering methods start with a single cluster, and then progressively splits or divides clusters.

We will focus on agglomerative methods.

9.3.1 Distance measures

Agglomerative clustering methods take the $n \times n$ matrix of inter-point distances $\mathbf{D} = (d_{ij})_{i,j=1}^{n}$ of the type we considered in Chapter 6. Sometimes we will have access to the underlying data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and in that case the distances may be computed using a distance function d, i.e., $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$. As we saw previously, we can find that different distance functions d, can produce different results. For example, with continuous data the Euclidean (L_2) and Manhattan distance (L_1) will often produce different results, with the Euclidean distance being more sensitive to outliers than the Manhattan distance:

- the Euclidean distance matrix for the points $x = 0, 1, 4$ is

$$\begin{pmatrix} 0 & & \\ 1 & 0 & \\ 16 & 9 & 0 \end{pmatrix}$$

whereas the Manhattan distance matrix is

$$\begin{pmatrix} 0 & & \\ 1 & 0 & \\ 4 & 3 & 0 \end{pmatrix}.$$

The differences can be even more important for binary attribute data. In Chapter 6 we saw the SMC and Jaccard similarity measures, which leads to a corresponding distance function. For example,

- The Jaccard distance for two sets of attributes A and B is

$$1 - \frac{|A \cap B|}{|A \cup B|}$$

i.e., it is number of attributes shared by A and B, divided by the number of attributes possessed by either A or B (or both), all subtracted from 1.

9.3.1.1 Distances between clusters

The distances between individual observations are the input to the clustering method. Each method then defines a distance between clusters. For example, suppose we have two clusters $\mathcal{G} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and $\mathcal{H} = \{\mathbf{x}_3, \mathbf{x}_4\}$. Clustering methods are characterized by how they measure the distance between clusters, $d(\mathcal{G}, \mathcal{H})$, which is a function of the pairwise distances d_{ij} where one member of the pair i is from $\mathcal{G}$ and the other, j, is from $\mathcal{H}$.

For example,

1. **single linkage (SL)** agglomerative clustering, sometimes called **nearest neighbour** clustering, uses the closest (least distant) pair

$$d_{SL}(\mathcal{G}, \mathcal{H}) = \min_{\substack{i \in \mathcal{G} \\ i \in \mathcal{H}}} d_{ij}$$

2. **complete linkage (CL)** agglomerative clustering, sometimes called **furthest neigbour** clustering, uses

$$d_{CL}(\mathcal{G}, \mathcal{H}) = \max_{\substack{i \in \mathcal{G} \\ i \in \mathcal{H}}} d_{ij}$$

3. **group average (GA)** clustering uses the average distance between groups:

$$d_{GA}(\mathcal{G}, \mathcal{H}) = \frac{1}{n_{\mathcal{G}} n_{\mathcal{H}}} \sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{H}} d_{ij}$$

and many other choices are possible (see the help page of the `hclust` command in R for some details).

Agglomerative clustering methods then work as follows:

- Start with the finest partition of singleton clusters (one observation per cluster)
- At each stage, join the two clusters with the closest distance.
- Stop once we are left with just a single cluster.

9.3.2 Toy Example

Suppose we are given 5 observations with distance matrix

```
(D <- as.dist(matrix(c(0,0,0,0,0,
                       2,0,0,0,0,
                       11,9,0,0,0,
                       15,13,10,0,0,
                       7,5,4,8,0), nr=5, byrow=T)))
```

```
##    1  2  3  4
## 2  2
## 3 11  9
## 4 15 13 10
## 5  7  5  4  8
```

i.e., $d_{1,2} = 2, d_{1,3} = 11$ etc.

All methods start with the singleton clusters $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$.

9.3.2.0.1 Single linkage

- at stage 1 we join the pair of clusters that are closest, which is $\{1\}$ and $\{2\}$, resulting in the clusters $\{1\,2\}, \{3\}, \{4\}, \{5\}$. The (single linkage) distance matrix for these clusters is then

```
##   12  3  4
## 3  9
## 4 13 10
## 5  5  4  8
```

- the closest clusters are $\{3\}$ and $\{5\}$, and so at stage 2 we get the clusters $\{1, 2\}, \{3, 5\}, \{4\}$, with the distances between clusters now given by

```
##    12 35
## 35  5
## 4  13  8
```

- at stage 3 we join $\{1, 2\}$ and $\{3, 5\}$ giving the clusters $\{1\,2, 3, 5\}, \{4\}$, with the distances between clusters now given by

```
##   1235
## 4    8
```

- at stage 4 we join the final two clusters resulting in the single cluster $\{1, 2, 3, 4, 5\}$.

The `hclust` command does agglomerative clustering: we just have to specify the method to use.

```
D.sl <-hclust(D, method="single")
```

We can use the `cutree` command to specify a given number of clusters. For example, it we want just two clusters, we can use

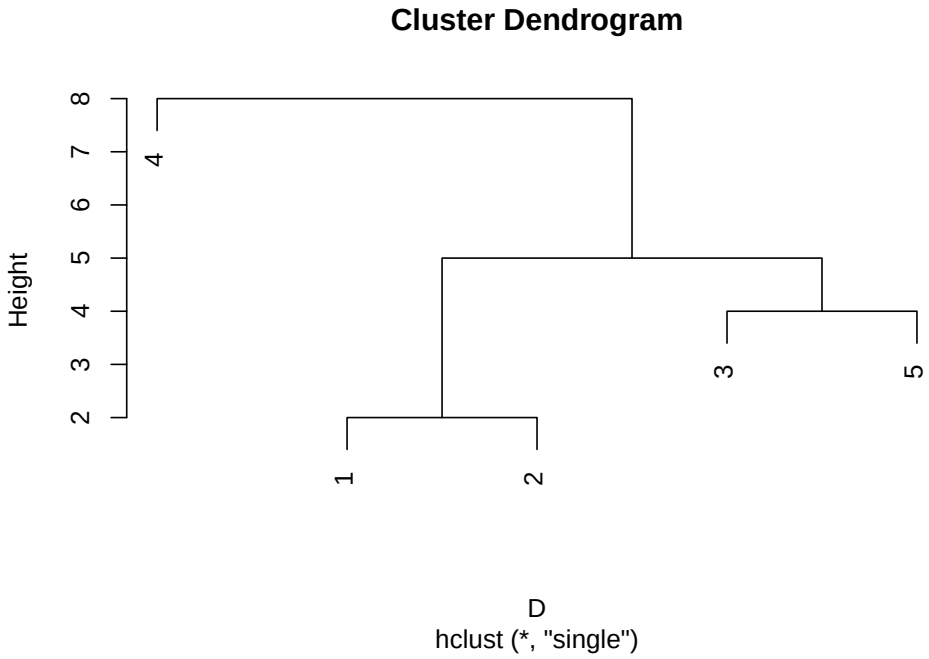
```
cutree(D.sl,k=2)
```

```
## [1] 1 1 1 2 1
```

This tells us that observations 1,2,3 and 5 are all in one cluster, and observation 4 in another cluster.

A convenient graphical way to present the output from an agglomerative clustering method is as a **dendrogram**. These show the arrangement of the clusters produced at each stage. The height of each node in the plot is proportional to the value of the intergroup distance between its two daughters. The `plot` command in R automatically produces dendrograms if passed output from the `hclust`.

```
plot(D.sl)
```



An alternatively to cutting the tree according to the number of clusters required, is to specify at which height to cut the tree. For example, if we cut the tree at height $T = 4.5$ we get

```
cutree(D.sl, h=4.5) # note we specify h= for height
```

```
## [1] 1 1 2 3 2
```

showing that we have clusters $\{1, 2\}, \{3, 5\}, \{4\}$.

9.3.2.0.2 Complete linkage

Let's now consider using complete linkage.

- at stage 0 the clusters are $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
- stage 1 is the same for all methods and we joint the pair of observations that are closest, resulting in clusters $\{1\,2\}, \{3\}, \{4\}, \{5\}$. The (complete linkage) distance matrix between clusters is

```
##    12  3  4
## 3  11
## 4  15 10
## 5   7  4  8
```

which is different to what we found previously.

- at stage 2 we joint together $\{3\}$ and $\{5\}$ giving the clusters $\{1, 2\}, \{3, 5\}, \{4\}$, with the distances between clusters given by

```
##    12 35
## 35 11
## 4  15 10
```

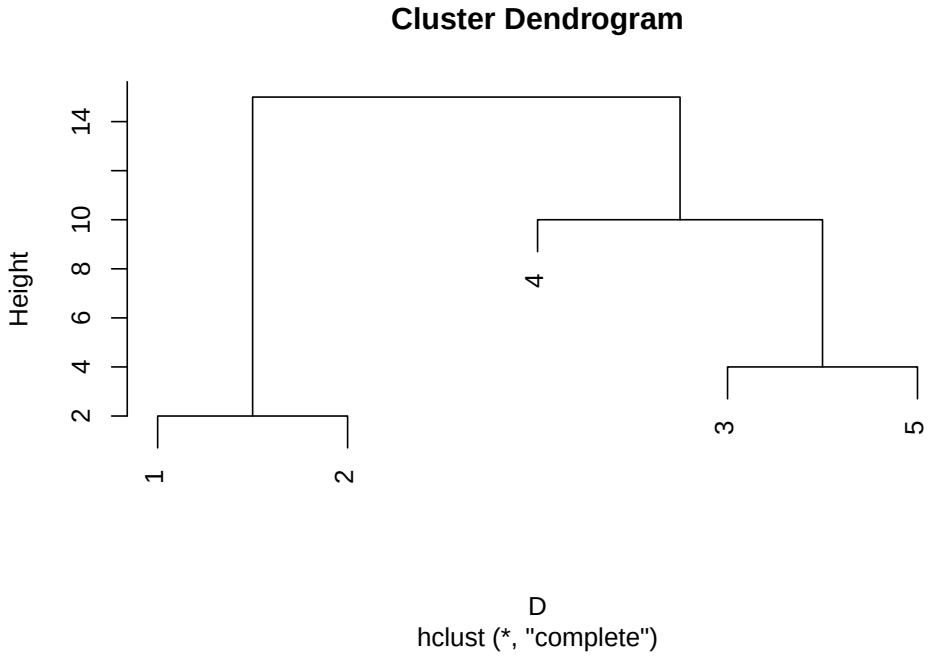
- at stage 3 the clusters are $\{1,2\},\{3,4,5\}$, with the distances between clusters given by

```
##     12
## 345 15
```

- at stage 4 there is a single cluster $\{1,2,3,4,5\}$.

The dendrogram for this is shown below. Note that complete linkage produces a different clustering if we require two clusters.

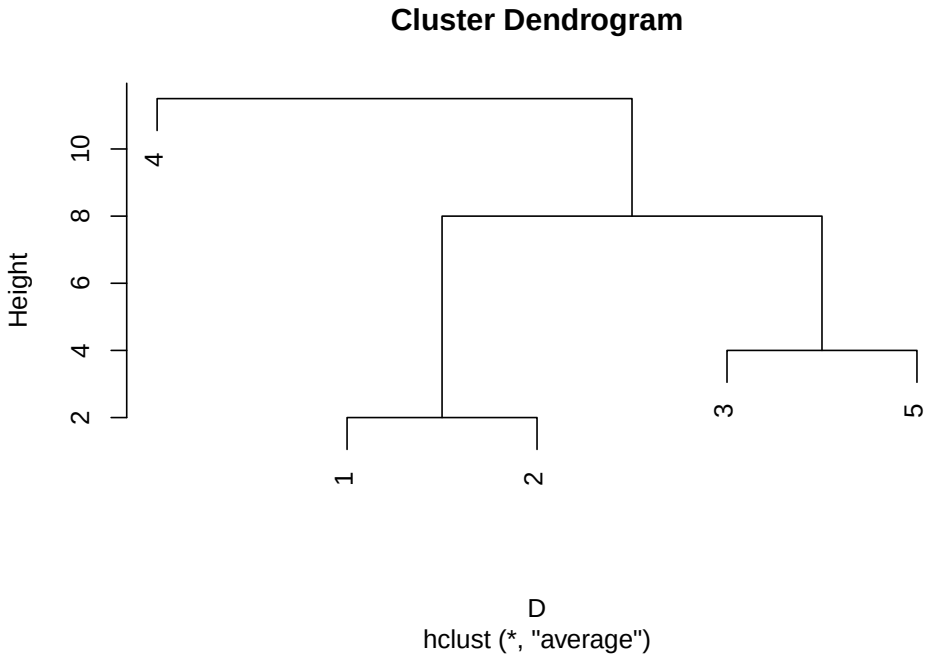
```
plot(hclust(D, method="complete"))
```



9.3.2.0.3 Group average

Group average clustering produces the same hierarchy of clusters as single linkage, but the nodes (points where clusters join) are at different heights in the dendrogram.

```
D.ga <- hclust(D, method="average")
plot(D.ga)
```



If we were to cut the single linkage and group average dendrograms at a height of 6, then we find different clusters. The single linkage dendrogram cut at height 6 gives the clusters $\{1, 2, 3, 5\}, \{4\}$ whereas the group average dendrogram gives the clusters $\{1, 2\}, \{3, 5\}, \{4\}$, despite the two trees having the same topology.

```
cutree(D.sl, h=6) # cut the tree at height 6
```

```
## [1] 1 1 1 2 1
```

```
cutree(D.ga, h=6)
```

```
## [1] 1 1 2 3 2
```

9.3.3 Comparison of methods

In the case where the distances $\mathbf{D}$ clearly split into distinct *compact* clusters (*compact* in the sense that all observations within a cluster are relatively close compared to observations in different clusters), all three methods will produce similar results.

In cases where the clustering is less obvious, we may find differences between the methods. Single linkage will join two clusters $\mathcal{G}$ and $\mathcal{H}$ if there is a pair of observations $i \in \mathcal{G}$ and $j \in \mathcal{H}$ that is small. We therefore get a chaining effect, where observations linked by a chain of close intermediate observations get clustered together. The consequence of this is that the clusters produced can end up not being *compact*. If we look at the largest distance between members

in a cluster, sometimes called the diameter,

$$D_{\mathcal{G}} = \max_{i,j \in \mathcal{G}} d_{ij}$$

then the single linkage method can produce clusters with very large diameters.

In constrast, complete linkage is the other extereme. Two clusters are only considered close if all pairs of obsevations are close. Consequently, it tends to produce clusters with small diameters. A downside is that observations assigned to a cluster can be much closer to members of other clusters than they are to some members of their own cluster.

Group averaging is a compromise between these two extremes. It aims to find compact clusters that are relatively far apart. However, the method is not invariant to monotone transformations of the distances, unlike single and complete linkage methods.

9.4 Summary

In this chapter we have focused on just two types of clustering methods: partitioning methods (K-means and model-based methods); and hierarchical agglomerative clustering methods.

There are many algorithms for performing Cluster Analysis, but there is no generally accepted *best* method. Different algorithms (or even the same algorithm with a different initialisation) do not necessarily produce the same results on a given dataset, and there is often a fairly large subjective element in the assessment of any particular method. Moreover, the choice of which underlying distance function to use can also lead to different conclusions.

One way to test a clustering algorithm is to apply it on data with a known group structure. Experience suggests that this will only produce good results when the groups are very distinct. When, on the other hand, there is a lot of overlap between groups, clustering algorithms are not likely to perform particularly well.

However, despite these cautionary remarks, clustering algorithms are often useful in practice, but it is an area where usually the most one can hope for is to find a good, but sub-optimal, solution.

9.5 Computer tasks

9.5.0.0.1 Q1

Perform hierarchical clustering with single, complete and average linkage using the iris data. You could also look at other method's such as Ward's method (see `?hclust` for details).

- In each case, cut the dendrogram to give three distinct groups, and compute the confusion matrix comparing the clusters found with the Species label. Commment on which linkage method has worked best in this case.
- We do not normally know the species/cluster-label when carrying out cluster analysis, and so can we still say anything about which methods are better if you were expecting to see three distinct groups?
- Compare the hierarchical clustering methods with the results of doing K-means clustering and model-based clustering (assuming multivariate normal distributions for each population).

9.5.0.0.2 Q2

Download the Indian Premier League data from Moodle and load it into R. We will filter the data to only look at players who played at least 10 innings, and the select just the information on the number of runs they scored, their high score (HS), their batting average (Avg), their best figures (BF), their strike rate (SR), and the number of 4s and 6s they hit.

```
library(dplyr)
IPL<-read.csv('IPL.csv')
IPL10<-IPL %>% dplyr::filter(Mat.x>=10) %>%
  dplyr::select(PLAYER,Runs.x, HS, Avg.x, BF, SR.x, X4s, X6s)
```

Apply agglomerative hierarchical clustering to these data. You will need to first compute a distance matrix for the data, which can be done with the `dist` command.

i. Using the Euclidean distance, do single linkage, complete linkage, and average linkage give similar dendrograms?

ii. Do your results change much if we use a different distance measure (e.g. Manhattan)?

iii. There are several R packages for creating different types of dendrogram plots. Have a look at the link here and try creating an alternative type of dendrogram. Consider whether this has helped communicate anything of interest about the data.

9.5.0.0.3 Q3

Look at the data stored in the `USArrests` data frame in R. You can read about the data by typing `?USArrests`.

Apply a selection of clustering methods to these data and discuss how many clusters appear to be present.

9.6 Exercises

1. In the table below, information is given on the presence or absence (denoted by 1 and 0, respectively) of 6 unspecified attributes, denoted A1, ..., A6, in Lions, Giraffes, Sheep and Humans.

Individual	A1	A2	A3	A4	A5	A6
Lion	1	1	0	0	1	1
Giraffe	1	1	1	0	0	1
Sheep	1	0	0	1	0	1
Human	0	0	0	0	1	0

   i. Use the information in the table to calculate a similarity matrix based on the simple matching coefficient given by Equation (6.10).

   ii. Using a suitable transformation, convert the similarity matrix into a dissimilarity matrix, $\mathbf{D}$ say.

   iii. Apply the single linkage method to the matrix $\mathbf{D}$. Summarise your results graphically in a dendrogram.

   iv. Apply the complete linkage method to the matrix $\mathbf{D}$. Summarise your results graphically in a dendrogram.

   v. Suppose exactly two clusters are required. What would be your clusters based on single linkage and complete linkage?

   vi. Repeat part v., but this time with three clusters rather than two.

   vii. Briefly (i.e. in one or two sentences) summarise your findings.

2. Attempt question 4b from the 2017-18 exam paper.

3. Prove Equation (9.5).

Chapter 10

Linear Models

In this chapter we consider linear regression methods. Suppose we have a response variable y that we would like to predict using covariate information $\mathbf{x} = (x_1, \ldots, x_p)^\top \in \mathbb{R}^p$.

A linear model for y assumes that the mean of y, $\mu = \mathbb{E}[y]$, is modelled as a linear function of $\mathbf{x}$, i.e. it is assumed that

$$\mu = \mathbf{x}^\top \boldsymbol{\beta},$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ is an unknown parameter vector to be estimated.

The videos for this chapter are available at the following links:

- Ordinary least squares
- Principal component regression
- Ridge regression
- Multi-output linear models

Notation

Suppose we have observations on n cases, $\{\mathbf{x}_1, y_1\}, \ldots, \{\mathbf{x}_n, y_n\}$. The standard linear model is

$$\begin{aligned} y_i &= \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i && (10.1) \\ &= \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i && (10.2) \end{aligned}$$

for $i = 1, \ldots, n$, where $\boldsymbol{\beta} \in \mathbb{R}^p$ is the unknown parameter vector we wish to estimate. We can write (10.1) in matrix form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad (10.3)$$

where $\overset{n\times p}{\mathbf{X}} = \begin{pmatrix} - & \mathbf{x}_1^\top & - \\ & \vdots & \\ - & \mathbf{x}_n^\top & - \end{pmatrix}$ is the matrix of covariates, $\overset{n\times 1}{\mathbf{y}}$ is the vector of univariate responses and $\overset{n\times 1}{\boldsymbol{\epsilon}}$ is the vector of univariate error terms.

We will assume throughout this chapter that $\mathbf{y}$ and $\mathbf{X}$ have been centred so that $\bar{y} = 0$ and the mean of each column of $\mathbf{X}$ is zero. We do this to simplify the exposition and to avoid having to include an intercept term in the model (which makes the notation slightly messy). For uncentred data, we can think of the model as

$$y_i - \bar{y} = \beta_1(x_{i1} - \bar{x}_1) + \ldots + \beta_p(x_{ip} - \bar{x}_p) + \epsilon_i$$

10.1 Ordinary least squares (OLS)

This section contains a brief review of the ordinary least squares (OLS) approach to fitting the linear model to data. Most of you will have studied this material in other modules.

Our model is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

In OLS we make three assumptionsabout the error term $\boldsymbol{\epsilon}$:

1. $\mathbb{E}[\epsilon_i] = 0$ for $i = 1, \ldots, n$.
2. The ϵ_i are uncorrelated, i.e. $\mathbb{C}\,\mathrm{ov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$.
3. The ϵ_i have constant variance (i.e., are *homoscedastic*): $\mathbb{V}\,\mathrm{ar}(\epsilon_i) = \sigma^2$ for all i.

If these assumptions are reasonable, then a common way to estimate $\boldsymbol{\beta}$ is to choose it to minimize the sum of squared errors, i.e.,

$$\begin{aligned} \hat{\boldsymbol{\beta}}^{ols} &= \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 \end{aligned}$$

If we note that

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

then multiplying out and differentiating with respect to $\boldsymbol{\beta}$ gives

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = 2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

You may need to look again at 2.1.4 to remind yourself about vector differentiation. Setting the derivative equal to $\mathbf{0}$ and solving gives the least squares estimator

$$\hat{\boldsymbol{\beta}}^{ols} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{10.4}$$

when $\mathbf{X}^\top \mathbf{X}$ is invertible (we'll discuss what to do if it is not in the next section). The **fitted-values** are

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}^{ols} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}.$$

10.1.1 Geometry

Does this make sense intuitively? As discussed in Chapter 2.3.3.2, we know that $\mathbf{X}\boldsymbol{\beta}$ is a linear combination of the columns of $\mathbf{X}$ and is thus in $\mathcal{C}(\mathbf{X})$, the column space of $\mathbf{X}$. Thus minimizing $||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$ is equivalent to finding the *orthogonal projection* of $\mathbf{y}$ onto $\mathcal{C}(\mathbf{X})$.

$$\mathbf{P}_{\mathcal{C}(X)}\mathbf{y} = \arg \min_{\mathbf{y}' : \mathbf{y}' \in \mathcal{C}(X)} ||\mathbf{y} - \mathbf{y}'||_2.$$

and by Proposition 2.5 this is

$$\mathbf{P}_{\mathcal{C}(X)}\mathbf{y} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} = \hat{\mathbf{y}}$$

which again implies

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

in agreement with what we found in the previous section. So we can think of linear regression as projecting $\mathbf{y}$ onto the columns of $\mathbf{X}$.

Its useful here to recall the exercise we looked at in Chapter 3 solving the linear system of equations. If the singular value decomposition (SVD) of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, then we can see that

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} \\
&= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top \mathbf{y} \\
&= \mathbf{X}^+ \mathbf{y}
\end{aligned}$$

i.e., our estimate of $\boldsymbol{\beta}$ is the generalized inverse of $\mathbf{X}$ formed by the SVD, $\mathbf{X}^+ = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top$, times $\mathbf{y}$.

The fitted values are

$$\begin{aligned}
\hat{\mathbf{y}}^{ols} &= \mathbf{X}\hat{\boldsymbol{\beta}} && (10.5) \\
&= \mathbf{U}\mathbf{U}^\top \mathbf{y} && (10.6) \\
&= \sum_{j=1}^{p} \mathbf{u}_j \mathbf{u}_j^\top \mathbf{y}. && (10.7)
\end{aligned}$$

$\mathbf{U}^\top \mathbf{y}$ are the coordinates of $\mathbf{y}$ with respect to the orthonormal basis $\mathbf{U}$, and $\mathbf{U}\mathbf{U}^\top \mathbf{y}$ is the projection of $\mathbf{y}$ onto that basis. So yet another way to think about OLS is that it projects $\mathbf{y}$ onto the left singular vectors $\mathbf{U}$. As we said in Chapter 3, the left singular vectors form an orthonormal basis for the column space of $\mathbf{X}$, and so projecting onto $\mathbf{U}$ is the same as projecting onto $\mathcal{C}(\mathbf{X})$.

10.1.2 Normal linear model

Often, we make an additional (fourth) assumption about the random errors and assume that they are normally distributed:

4. The ϵ_i are IID $N(0, \sigma^2)$.

The log-likelihood for models (10.1) and (10.3) under the Gaussian assumption 4. is

$$\begin{aligned}\ell(\boldsymbol{\beta}, \sigma^2) &= -\frac{n}{2}\log(\sigma^2) - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 \\ &= -\frac{n}{2}\log(\sigma^2) - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).\end{aligned}$$

We can see that maximizing the log-likelihood with respect to $\boldsymbol{\beta}$ is equivalent to minimizing $||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$ and thus $\hat{\boldsymbol{\beta}}^{ols}$ is also the maximum likelihood estimator of $\boldsymbol{\beta}$ when the errors have a Gaussian distribution.

10.1.3 Linear models in R

Linear models are easy to fit in R. For example, using the iris data, we can train a model to predict the sepal length from the other measurements as follows:

```
lm(Sepal.Length~ Sepal.Width+Petal.Length+Petal.Width,data=iris)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
##     data = iris)
##
## Coefficients:
##  (Intercept)   Sepal.Width  Petal.Length   Petal.Width
##       1.8560        0.6508        0.7091       -0.5565
```

10.1.4 Problems with OLS

There are several problems encountered when fitting linear models using OLS when working with large datasets, two of which are

- $\mathbf{X}^\top \mathbf{X}$ may not be invertible, or the inversion may be numerically unstable (which is often called *multicollinearity*). For example, if the number of

covariates p is larger than the number of observations n (which can easily happen when working with image data), then the $p \times p$ matrix $\mathbf{X}^\top \mathbf{X}$ is at most rank $n < p$ and so is not invertible.

- Prediction accuracy may be low. The Gauss Markov theorem says that the ordinary least squares estimator of $\boldsymbol{\beta}$ has minimum variance amongst all unbiased estimators of $\boldsymbol{\beta}$. The mean square error (MSE), which is a measure of accuracy, of an estimator $\tilde{\theta}$ of unknown quantity θ is

$$\begin{aligned} MSE(\tilde{\theta}) &= \mathbb{E}((\theta - \tilde{\theta})^2) \\ &= \mathbb{V}\,\mathrm{ar}(\tilde{\theta}) + (\mathbb{E}(\tilde{\theta}) - \theta)^2 \\ &= \mathbb{V}\,\mathrm{ar}(\tilde{\theta}) + \mathrm{bias}(\tilde{\theta})^2 \end{aligned}$$

  By allowing a small amount of bias, we may be able to find estimators that have a smaller MSE than the least squares estimator of $\boldsymbol{\beta}$.

In the next two sections we will look at some methods that have been proposed to solve these problems.

10.2 Principal component regression (PCR)

In **principal component regression**, instead of regressing $\mathbf{y}$ onto $\mathbf{X}$ (or equivalently projecting $\mathbf{y}$ onto the columns of $\mathbf{X}$), we instead regress $\mathbf{y}$ onto the principal component scores of $\mathbf{X}$. If the singular value decomposition (SVD) of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ (recall that $\mathbf{X}$ is column centred), then we've seen that

$$\hat{\boldsymbol{\beta}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top \mathbf{y}.$$

Note that this does not rely upon $\mathbf{X}^\top \mathbf{X}$ being invertable. If $\mathbf{X}$ is rank r, then it has r non-zero singular values, and $\mathbf{V}$ is $p \times r$ and $\mathbf{U}$ is $n \times r$. In principal component regression we take this idea further.

We start by projecting the covariates matrix $\mathbf{X}$ onto the first k principal components of $\mathbf{X}$, giving us the first k principal component scores of $\mathbf{X}$:

$$\mathbf{Z} = \mathbf{X}\mathbf{V}_k = \mathbf{U}_k\boldsymbol{\Sigma}_k$$

where $\mathbf{U}_k = \begin{pmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_k \end{pmatrix} \in \mathbb{R}^{n \times k}$ is the matrix formed from the first k columns of $\mathbf{U}$, and $\mathbf{V}_k \in \mathbb{R}^{p \times k}$ is the matrix formed from the first k columns of $\mathbf{V}$, and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{k \times k}$ is the diagonal matrix containing the first k singular values (cf Section 3.5).

In PCR, we then use the linear model

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\epsilon}.$$

Estimating $\boldsymbol{\alpha}$ by least squares gives

$$\begin{aligned}\hat{\boldsymbol{\alpha}} &= (\mathbf{Z}^\top \mathbf{Z})^{-1}\mathbf{Z}^\top \mathbf{y} \\ &= (\boldsymbol{\Sigma}_k \mathbf{U}_k^\top \mathbf{U}_k \boldsymbol{\Sigma}_k)^{-1}\boldsymbol{\Sigma}_k \mathbf{U}_k^\top \mathbf{y} \\ &= \boldsymbol{\Sigma}_k^{-1}\mathbf{U}_k^\top \mathbf{y}\end{aligned}$$

as $\mathbf{U}_k^\top \mathbf{U}_k = \mathbf{I}_k$. The corresponding fitted values will be

$$\hat{\mathbf{y}}^{PCR} = \mathbf{Z}\hat{\boldsymbol{\alpha}} \tag{10.8}$$

$$= \mathbf{U}_k \mathbf{U}_k^\top \mathbf{y} \tag{10.9}$$

$$= \sum_{j=1}^{k} \mathbf{u}_k \mathbf{u}_j^\top \mathbf{y}. \tag{10.10}$$

Comparing with Equation (10.7), we can see this is the same as before, except we have truncated the sum, so that we are only projecting onto the first k principal components. If we let $k = r = \text{rank}(\mathbf{X})$ then PCR is the same as OLS.

To convert from the PCR coefficients $\boldsymbol{\alpha}$ back to coefficients of the original $\mathbf{x}$ variables, we multiply by the right singular vectors:

$$\hat{\boldsymbol{\beta}}^{pcr} = \mathbf{V}_k \hat{\boldsymbol{\alpha}} = \mathbf{V}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{U}_k^\top \mathbf{y}$$

as $\mathbf{Z}\boldsymbol{\alpha} = \mathbf{X}\mathbf{V}_k\boldsymbol{\alpha}$.

We choose the number of principal components to use in the regression k, by assessing some form of predictive accuracy of the resulting model.

The are several motivations for using PCR. The first is that it can always be used, regardless of whether $\mathbf{X}^\top\mathbf{X}$ is invertible. The second is that in many problems with large p, it often has superior predictive performance to OLS, as it reduces the variance of the predictions.

10.2.1 PCR in R

PCR is easy to implement yourself in R. For example, using the iris regression problem from the previous section, we can do PCR using just the first 2 principal components as follows:

```
iris.pca <- prcomp(iris[,2:4],scale=TRUE)
Z = iris.pca$x[,1:2] # select the first two PCs
iris.lm <- lm(iris$Sepal.Length~Z)
iris.lm
```

```
##
## Call:
```

```
## lm(formula = iris$Sepal.Length ~ Z)
##
## Coefficients:
## (Intercept)          ZPC1          ZPC2
##      5.8433        0.4230       -0.4348
```

Note that we didn't centre the data here, and so we have estimate a non-zero intercept as well as the coefficients of $\mathbf{x}$.

To convert from PCR coefficients to coefficients of $\mathbf{x}$ we can multiply by $\mathbf{V}_k$

```
iris.pca$rotation[,1:2] %*% coef(iris.lm)[-1]
```

```
##                    [,1]
## Sepal.Width   0.2173767
## Petal.Length 0.3853085
## Petal.Width   0.4150270
```

In practice we do not need to do PCR ourselves in this way, as it has been implemented in the `pls` R package.

```
library(pls)
iris.pcr <- pcr(Sepal.Length~ Sepal.Width+Petal.Length+Petal.Width, 2, # number of components,
                scale=TRUE, data=iris)
coef(iris.pcr)
```

```
## , , 2 comps
##
##               Sepal.Length
## Sepal.Width      0.2173767
## Petal.Length     0.3853085
## Petal.Width      0.4150270
```

Note that the coefficients found match those we computed ourselves.

To choose the number of components to retain, we can use cross-validation. This is a generalization of the idea of splitting the data into training and test sets. The `pls` package reports the estimate root means square error on the test set (RMSEP) for each possible value of k, up to the maximum specified by the user (taken to be the rank of $\mathbf{X}$ if we leave it unspecified).

```
iris.pcr2 <- pcr(Sepal.Length~ Sepal.Width+Petal.Length+Petal.Width, 3, # number of components,
                scale=TRUE, data=iris, validation='CV')
summary(iris.pcr2)
```

```
## Data:    X dimension: 150 3
##  Y dimension: 150 1
## Fit method: svdpc
## Number of components considered: 3
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps
## CV          0.8308   0.5476   0.3926   0.3215
## adjCV       0.8308   0.5470   0.3921   0.3209
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps
## X               74.05    98.84   100.00
## Sepal.Length    57.97    78.47    85.86
```

10.3 Shrinkage methods

Another approach to fitting linear models is to regularise the parameters. Instead of minimizing $||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$, we instead minimize

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||$$

where we have added a penalty term $\lambda||\boldsymbol{\beta}||$ to constrain the size of the parameter ($\lambda \geq 0$). This stops estimated values of $\boldsymbol{\beta}$ becoming too large. Larger values of λ penalise the size of $\boldsymbol{\beta}$ more strongly, and so have a bigger effect. As $\lambda \to \infty$ we find the optimal $\boldsymbol{\beta}$ tends to $\mathbf{0}$.

Ridge regression is the name given to the estimate found when we use the $||\cdot||_2$ norm.

$$\hat{\boldsymbol{\beta}}^{ridge} = \arg\min_{\boldsymbol{\beta}} \left\{ ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2 \right\}.$$

You'll show on the example sheets that

$$\hat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

We can also show that the ridge regression estimator is small than the OLS estimator

$$||\hat{\boldsymbol{\beta}}^{ols}||_2 \geq ||\hat{\boldsymbol{\beta}}^{ridge}||_2$$

We say that ridge regression *shrinks* the parameter estimates towards zero. The motivation for this is that when there are many correlated covariates, we sometimes find that the coefficients are poorly determined, and so the OLS estimator can have high variance. As Hastie et al. put it

> "A wildly large positive coefficient on one variable can be cancelled by a large negative coefficient on its correlated cousin. By imposing a size constraint on the coefficients [...] this phenomenon is prevented from occurring."

The fitted values are again $\hat{\mathbf{y}}^{ridge} = \mathbf{X}\hat{\boldsymbol{\beta}}^{ridge}$. If we substitute the SVD for $\mathbf{X}$ we find

$$\begin{aligned}
\hat{\mathbf{y}}^{ridge} &= \mathbf{X}\hat{\boldsymbol{\beta}}^{ridge} \\
&= \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top(\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top + \lambda\mathbf{I})^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y} \\
&= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top(\mathbf{V}(\boldsymbol{\Sigma}^2 + \lambda\mathbf{I})\mathbf{V}^\top)^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y} \\
&= \mathbf{U}\boldsymbol{\Sigma}(\boldsymbol{\Sigma}^2 + \lambda\mathbf{I})^{-1}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y} \\
&= \sum_{i=1}^{p} \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\mathbf{u}_i\mathbf{u}_i^\top\mathbf{y}
\end{aligned}$$

The penultimate line has assumed $\mathbf{V}$ is full rank here, but the argument can be made to work regardless. Since $\lambda \geq 0$, we have that $\frac{\sigma_i^2}{\sigma_i^2+\lambda} \leq 1$. So we can see that just like in OLS and PCR, we project the data onto the orthonormal basis $\mathbf{U}$, but with ridge regression, we shrink the coefficients by the factors $\frac{\sigma_i^2}{\sigma_i^2+\lambda} \leq 1$. A greater amount of shrinkage is applied to basis vectors with small singular values σ_i.

If we compare the three different expressions for the fitted values

$$\begin{aligned}
\hat{\mathbf{y}}^{ols} &= \sum_{i=1}^{p} \mathbf{u}_i\mathbf{u}_i^\top\mathbf{y}, \\
\hat{\mathbf{y}}^{ridge} &= \sum_{i=1}^{k} \mathbf{u}_i\mathbf{u}_i^\top\mathbf{y} \\
\hat{\mathbf{y}}^{ridge} &= \sum_{i=1}^{p} \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\mathbf{u}_i\mathbf{u}_i^\top\mathbf{y}
\end{aligned}$$

we can see that PCR just works in a lower dimensional space discarding dimensions with small singular values, whereas ridge regression retains all coordinates, but reduces the importance of the coordinates with small singular values.

In ridge regression we used the $||\cdot||_2$ norm to penalise the parameter size. Other methods use other norms, which can induce interesting effects. Of particular interest is the **lasso**, which uses the L_1 norm as a penalty $||\boldsymbol{\beta}||_1$. This induces sparsity in the solution, but is beyond the scope of the module. If you are interested, take a look at the wikipedia page

10.3.1 Ridge regression in R

We'll use the `glmnet` package to do ridge regression. The `glmnet` command solves the optimization problem

$$\arg\min_{\boldsymbol{\beta}} \left\{ ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda \left((1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1^2 \right) \right\}$$

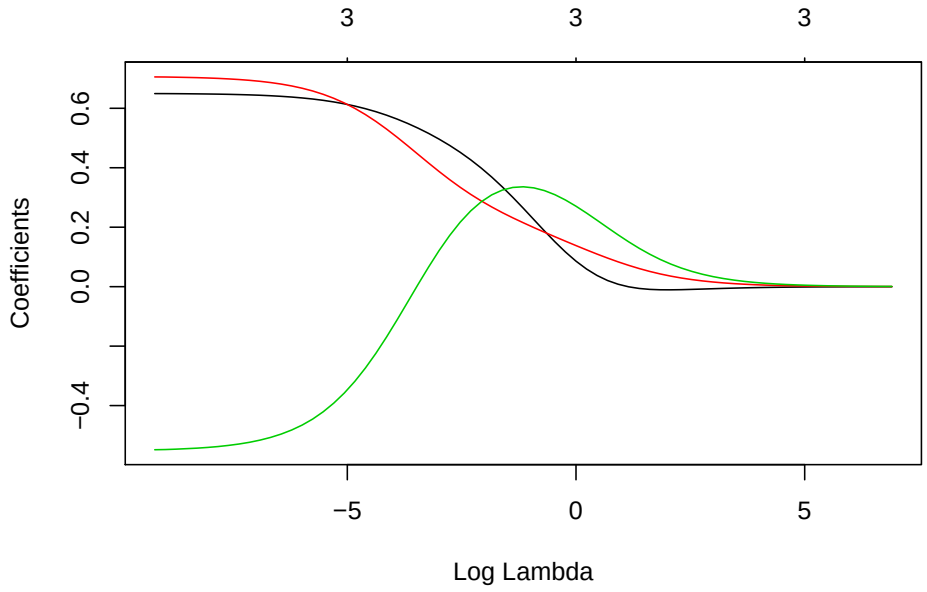
where we must specify the value of α. Taking $\alpha = 0$ gives us ridge regression, and $\alpha = 1$ gives the lasso. Values of $\alpha \in (0, 1)$ balance both the L_1 and L_2 penalties (a method known as the **elastic-net**).

We can choose to specify what values of λ to use (if we don't specify this, the `glmnet` package will pick some sensible values). Remember that larger values of λ result in a bigger penalty, and greater shrinkage of the parameters.

```
library(glmnet)
X = as.matrix(iris[,2:4]) # glmnet doesn't work with data frames
y=iris[,1]
lambdas <- 10^seq(3, -4, by = -.1)
iris.ridge <- glmnet(X,y, alpha=0, lambda = lambdas)
```

We can then look at how the parameter estimates change as the size of λ changes.

```
plot(iris.ridge, xvar='lambda')
```




```
#matplot(log(lambdas), t(iris.ridge$beta), type='l') # does the same
```

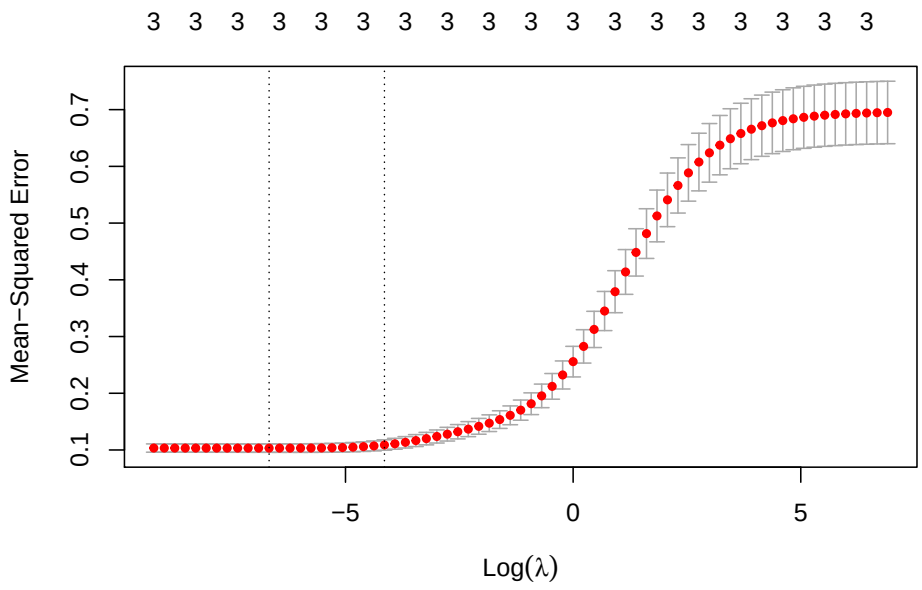
We can see that as λ grows large, the parameters all shrink to 0 as expected. For $\lambda \approx 0$, we get the OLS estimates of the parameters

```
coef(lm(y~X))
```

```
##   (Intercept)  XSepal.Width XPetal.Length  XPetal.Width
##     1.8559975     0.6508372     0.7091320    -0.5564827
```

We usually choose λ by finding which value gives the lower prediction error (using some form of predictive test, such as cross-validation).

```
cv_fit <- cv.glmnet(X, y, alpha = 0, lambda = lambdas)
plot(cv_fit)
```



From this, we can see that very small values of λ work best here, i.e., the OLS estimator is optimal. This is not a surprise as there are only three covariates in this problem. It is in larger problems that we expect shrinkage methods to be most useful.

The value of lambda that minimizes the prediction error can be found by

```
cv_fit$lambda.min
```

```
## [1] 0.001258925
```

and the largest value of lambda that gives a prediction error within one standard deviation of the minimum can be found using:

```
cv_fit$lambda.1se
```

```
## [1] 0.01584893
```

10.4 Multioutput Linear Model

In the standard linear model the responses y_i are univariate. In the multivariate linear model, the data are $\{\mathbf{x}_1, \mathbf{y}_1\}, \ldots, \{\mathbf{x}_n, \mathbf{y}_n\}$ with the responses $\mathbf{y}_i \in \mathbb{R}^q$. Here, the linear model takes the form

$$\mathbf{y}_i = \mathbf{B}^\top \mathbf{x}_i + \boldsymbol{\epsilon}_i, \qquad i = 1, \ldots, n, \tag{10.11}$$

where $\mathbf{B}$ is now a $p \times q$ parameter matrix, and we have an error vector $\boldsymbol{\epsilon}_i \in \mathbb{R}^q$ are $q \times 1$. The model (10.11) may be written in matrix form as

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E}, \tag{10.12}$$

where

$$\mathbf{Y} = \begin{pmatrix} - & \mathbf{y}_1^\top & - \\ & \vdots & \\ - & \mathbf{y}_n^\top & - \end{pmatrix}$$

is the $n \times q$ data matrix for the y-variables, $\mathbf{X}$ is the $n \times p$ data matrix as defined before, and

$$\underset{n \times p}{\mathbf{E}} = \begin{pmatrix} - & \boldsymbol{\epsilon}_1^\top & - \\ & \vdots & \\ - & \boldsymbol{\epsilon}_n^\top & - \end{pmatrix}.$$

We will assume that

$$\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n \quad \text{are IID} \quad N_p(\mathbf{0}_p, \boldsymbol{\Sigma}). \tag{10.13}$$

Proposition 10.1. *The maximum likelihood estimator of* $\mathbf{B}$ *is*

$$\hat{\mathbf{B}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \tag{10.14}$$

Remarks:

1. note how similar (10.14) is to its univariate counterpart (10.4). The only thing that is different is that $\mathbf{Y}$ is now an $n \times q$ matrix rather than an $n \times 1$ vector.
2. The coefficients for the k^{th} output, i.e., the k^{th} column of $\mathbf{B}$, are just the OLS estimates from regressing the k^{th} column of $\mathbf{Y}$ on $\mathbf{X}$. I.e., the multivariate linear regression estimates are just the univariate estimates applied to each output in turn: the different outputs do not affect each others least squares estimates.

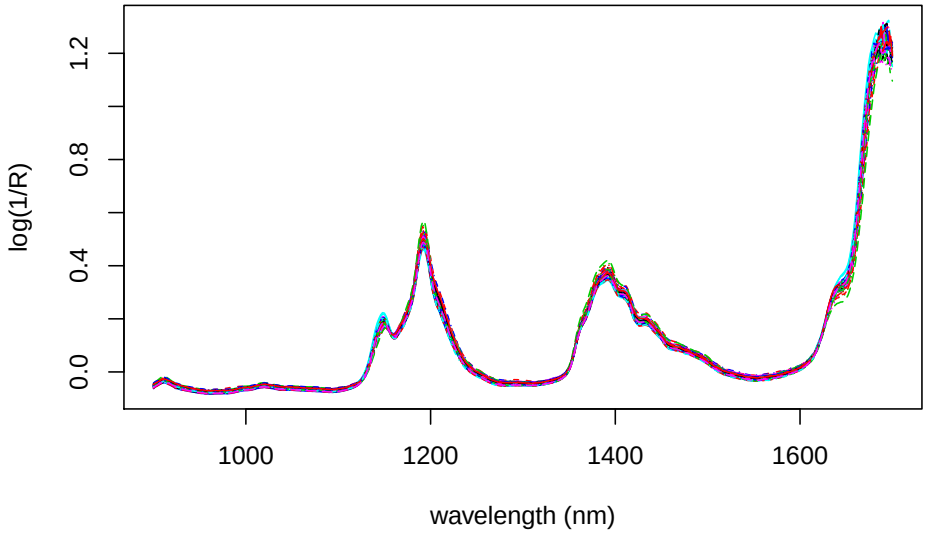
You will see how to fit multivariate linear models in R in the computer tasks.

10.5 Computer tasks

10.5.0.0.1 Q1

The gasoline dataset in the `pls` dataset consists of data on 60 different gasoline samples. For each sample, the near-infra-read (NIR) spectrum has been measured, which consists of measurements at 401 different wavelengths.

```
library(pls)
data(gasoline)
matplot(seq(900,1700,2),t(gasoline$NIR), type='l', ylab='log(1/R)', xlab='wavelength (nm)')
```



Also recorded is the octane number of each sample. We will use the NIR measurements to predict the octane number of each sample. Note that here $n = 60$ and $p = 401$, so we have many more covariates than samples.

i. Start by using the `lm` command to predict the octane number from the NIR spectra.

```
lm(octane~NIR, data=gasoline)
```

What happens? How many of the coefficients are estimated?

ii. We'll now try principal component regression. First, project the NIR values onto the first 10 principal components. Then build a linear model using `lm`. Split the data into a training set containing 40 observations, and a test set containing 20 observations. What is the predictive accuracy of your model?

iii. Now use the `pcr` command from the `pls` package. You can use cross-validation to assess the prediction accuracy, and plot the predictive accuracies, for different number of principal components using the code below.

How many principal components would you retain in your model to ensure the best performance?

```
gasoline.pcr <- pcr(octane~NIR, data=gasoline, validation='CV')
summary(gasoline.pcr)
plot(RMSEP(gasoline.pcr), legendpos = "topright")
```

iv. Now try ridge regression. What value would you choose for the ridge penalty term λ?

10.5.0.0.2 Q2

The `lm` command in R will automatically work if you pass in a matrix of response variables. To demonstrate this, lets try to predict the sepal width and sepal length from the petal length and widths for the iris data.

```
lm(cbind(Sepal.Length, Sepal.Width) ~ Petal.Length+Petal.Width, data=iris)
```

Check that the parameter estimates you find are the same as if you did two univariate linear regressions.

10.6 Exercises

1. Show that the ridge regression estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}.$$

   Prove that the inverse exists if $\lambda > 0$.

   Finally, prove that the estimator can rewritten as

$$\hat{\boldsymbol{\beta}}^{ridge} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y}.$$

   When might this be useful?

2. Let

$$\hat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda ||\boldsymbol{\beta}||$$

   Prove that

$$||\hat{\boldsymbol{\beta}}_\lambda|| \leq ||\hat{\boldsymbol{\beta}}^{ols}||.$$

   Note that we can prove this for a general norm $||\cdot||$, not just the L_2 norm.

3. For the normal linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + N(\mathbf{0}, \sigma^2 \mathbf{I})$$

   show that

$$\mathbb{V}\text{ar}(\hat{\boldsymbol{\beta}}^{ols}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1} \tag{10.15}$$

$$= \sigma^2 \sum_{i=1}^{p} \frac{\mathbf{v}_i \mathbf{v}_i^\top}{\lambda_i} \tag{10.16}$$

where λ_i are the eigenvalues of $\mathbf{X}^\top\mathbf{X}$, and $\mathbf{v}_i$ the corresponding unit eigenvectors.

If $\hat{\boldsymbol{\beta}}_k$ is the PCR estimator based on the first k principal components, show that

$$\mathbb{V}\,\mathrm{ar}(\hat{\boldsymbol{\beta}}_k) = \sigma^2 \sum_{i=1}^{k} \frac{\mathbf{v}_i \mathbf{v}_i^\top}{\lambda_i}.$$

Thus, show that

$$\mathbf{A}_k = \mathbb{V}\,\mathrm{ar}(\hat{\boldsymbol{\beta}}^{ols}) - \mathbb{V}\,\mathrm{ar}(\hat{\boldsymbol{\beta}}_k)$$

is a positive semi-definite matrix. Consequently, show that any linear combination of $\hat{\boldsymbol{\beta}}_k$, e.g., a prediction $\mathbf{x}^\top\hat{\boldsymbol{\beta}}_k$, has a lower variance compared to the same linear combination using the OLS estimator, i.e.,

$$\mathbb{V}\,\mathrm{ar}(\mathbf{x}^\top\hat{\boldsymbol{\beta}}_k) \leq \mathbb{V}\,\mathrm{ar}(\mathbf{x}^\top\hat{\boldsymbol{\beta}}^{ols}).$$