Multivariate Statistics with **R**

Paul J. Hewson

March 17, 2009

Contents

Books

Many of the statistical analyses encountered to date consist of a *single response variable* and *one or more explanatory variables*. In this latter case, *multiple regression*, we regressed a single response (dependent) variable on a number of explanatory (independent) variables. This is occasionally referred to as "multivariate regression" which is all rather unfortunate. There isn't an entirely clear "canon" of what is a multivariate technique and what isn't (one could argue that discriminant analysis involves a single dependent variable). However, we are going to consider the simultaneous analysis of a number of related variables. We may approach this in one of two ways. The first group of problems relates to classification, where attention is focussed on individuals who are more alike. In unsupervised classification (cluster analysis) we are concerned with a range of algorithms that at least try to identify individuals who are more alike if not to distinguish clear groups of individuals. There are also a wide range of scaling techniques which help us visualise these differences in lower dimensionality. In supervised classification (discriminant analysis) we already have information on group membership, and wish to develop rules from the data to classify future observations.

The other group of problems concerns inter-relationships between variables. Again, we may be interested in lower dimension that help us visualise a given dataset. Alternatively, we may be interested to see how one group of variables is correlated with another group of variables. Finally, we may be interested in models for the interrelationships between variables.

This book is still a work in progress. Currently it contains material used as notes to support a module at the University of Plymouth, where we work in conjunction with Johnson and Wichern (1998). It covers a reasonably established range of multivariate techniques. There isn't however a clear "canon" of multivariate techniques, and some of the following books may also be of interest: Other Introductory level books:

- Afifi and Clark (1990)
- Chatfield and Collins (1980)
- Dillon and Goldstein (1984)
- Everitt and Dunn (1991)

- Flury and Riedwyl (1988)
- Johnson (1998)
- Kendall (1975)
- Hair et al. (1995)
- et al. (1998)
- Manly (1994)

Intermediate level books:

- Flury (1997) (My personal favourite)
- Gnanadesikan (1997)
- Harris (1985)
- Krzanowski (2000) ?Krzanowski and Marriott (1994b)
- Rencher (2002)
- Morrison (2005)
- Seber (1984)
- Timm (1975)

More advanced books:

- Anderson (1984)
- Bilodeau and Brenner (1999)
- Giri (2003)
- Mardia et al. (1979)
- Muirhead (York)
- Press (1982)
- Srivastava and Carter (1983)

Some authors include contingency tables and log-linear modelling, others exclude Cluster analysis. Given that multivariate methods are particularly common in applied areas such Ecology and Psychology, there is further reading aimed at these subjects. It is quite possible that they will have very readable descriptions of particular techniques.

Whilst this book is still an alpha-version work in progress, the aim is

(a) To cover a basic core of multivariate material in such a way that the core mathematical principles are covered

(b) To provide access to current applications and developments

There is little material included yet for (b) (although sketch notes are being worked on). Comments, feedback, corrections, co-authors are all welcome.

Chapter 1

Multivariate data

1.1 The nature of multivariate data

We will attempt to clarify what we mean by multivariate analysis in the next section, however it is worth noting that much of the data examined is observational rather than collected from designed experiments. It is also apparent that much of the methodology has been developed outside the statistical literature. Our primary interest will be in examining continuous data, the only exception being categorical variables indicating group membership. This may be slightly limiting, but we will also tend to rely on at least asymptotic approximations to (multivariate) normality, although these are not always necessary for some techniques. The multivariate normal distribution is a fascinating subject in its own right, and experience (supplemented with some brutal transformations) indicates it is a reasonable basis for much work. Nevertheless, there is considerable interest in robust methods at the moment and we refer to some of theseapproaches where possible.

1.2 The role of multivariate investigations

If we assume that linear and generalised linear models (and their descendants) are the mainstay of statistical practice, there is a sense in which most statistical analysis is multivariate. However, *multivariate analysis* has come to define a canon of methods which could be characterised by their use of the dependence structure between a large number of variables. This canon has not yet been firmly established; we attempt one definition of it here but omit some methods others would include and include some methods others would omit. We would suggest that multivariate analysis has either the units as a primary focus, or involves an assessment primarily of the variables. When considering the units, we usually refer to techniques for classification; supervised classfication if we

already understand the grouping and unsupervised classification where we have no *a priori* knowledge of any groupings within our observed units. The multivariate methodology at the core of supervised classification is discriminant analysis, although the machine learning community has developed many other approaches to the same task. We will consider these techniques in the light of hypothesis tests (Hotelling's T^2 test and Multivariate Analysis of Variance) which might help us determine whether groupings within our data really are distinct. Unsupervised classification has traditionally been associated with cluster analysis, a wide range of algorithms which attempt to find structure in data. It is perhaps cluster analysis that is the most often contested component of our multivariate canon - some authorities prefer approaches based less on automated algorithms and rather more on statistical models and would argue for approaches such as mixture models and perhaps latent class analysis. Given the reliance of cluster analysis on distance measures, we will also consider scaling techniques as a method of visualising distnace.

In considering the relationship between variables, we will spend some time exploring principal components, the most misused of all multivariate techniques which we consider primarily as a projection technique. Some coverage will also be given to canonical correlation, an attempt to understand the relationship between two sets of variables. Finallly, we will consider factor analysis, a much contested technique in statistical circles but a much used one in applied settings.

In order to make some sense of these techniques, we will present a brief overview of linear algebra as it pertains to the techniques we wish to explore, and will present some properties of the multivariate normal distribution.

1.3 Summarising multivariate data (presenting data as a matrix, mean vectors, covariance matrices

A number of datasets will be used thoughout the course, where these are not available within R itself they will be posted in the student portal. For now, consider the *USArrests* data. This was published by McNeil, D. R. (1977) "Interactive Data Analysis", Wiley, and gives Arrest rates in 1973 (derived from World Almanac and Book of facts 1975. and Urban population rates derived from Statistical Abstracts of the United States 1975. We therefore consider data on "Murder" (arrests per 100,000), Assault (arrests per 100,000), Rape (arrests per 100,000) and the percentage of the population living in urban areas in each state.

1.3.1 Data display

A matrix is a convenient way of arranging such data.

....................*State* *Murder* *Assault* *Rape* *UrbanPop*(%)

Alabama	13.2	236	21.2	58
Alaska	10.0	263	44.5	48
Arizona	8.1	294	31.0	80
Arkansas	8.8	190	19.5	50
California	9.0	276	40.6	91
Colorado	7.9	204	38.7	78
Connecticut	3.3	110	11.1	77
Delaware	5.9	238	15.8	72
Florida	15.4	335	31.9	70
Georgia	17.4	211	25.8	60
Hawaii	5.3	46	20.2	83
...	...	...	...	

Note in total that there are 50 states, (this display had been cut off after the 11th row, Hawaii), and that there are four variables. Have a look at the USArrests data itself, and the associated helpfile:

```
> ?USArrests
> summary(USArrests)
> USArrests
```

1.4 Graphical and dynamic graphical methods

1.4.1 Chernoff's Faces

One of the more charismatic ways of presenting multivariate data was proposed by Chernoff, H. (1973) "The use of faces to represent statistical association", JASA, 68, pp 361-368. (see www.wiwi.uni-bielefeld.de/ wolf/ for the R code to create these). If you have loaded the mvmmisc.R file, you can get these by typing:

```
> faces(USArrests}
```

However, there are more useful ways of investigating multivariate data. Slightly less wild, there are star plots, which depict the data as beams. There are as many beams as there are variables, and the length of the beam reflects the value of the variable.

```
> stars(USArrests)
```

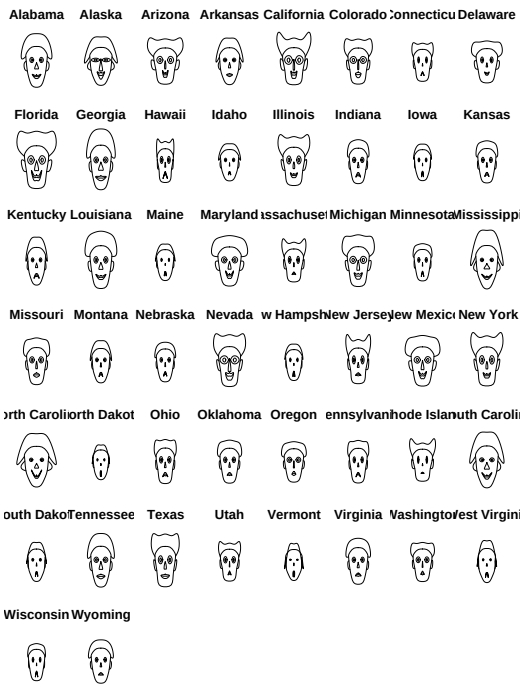


Figure 1.1: US Arrest data presented as Chernoff's faces

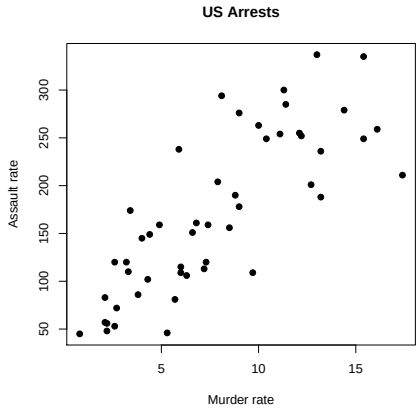


Figure 1.2: Scatter plot of Murder rate against Assault rate for US States in 1973

1.4.2 Scatterplots, pairwise scatterplots (draftsman plots)

Scatterplots should already be familiar as a means of exploring the relationship between two variables.

```
> attach(USArrests)
> plot(Murder, Assault)
> par(las = 1) ## Horizontal axis units on y axis
> plot(Murder, Assault, main = "Scatter plot", pch = 16)
> detach(USArrests)
```

However, we have more than two variables of interest. A set of pairwise scatterplots (sometimes called a draftsman plot) may be of use:

```
> pairs(USArrests, pch = 16)
```

There other useful functions available. For example what does `splom` do? (Look up `>?splom`).

```
> library(lattice)
> splom(~USArrests)
```

1.4.3 Optional: 3d scatterplots

This bit is optional: feel free to have a go if you want to find out about installing R libraries.

There are facilities in R for making 3d effect scatterplots: you need to download and install an

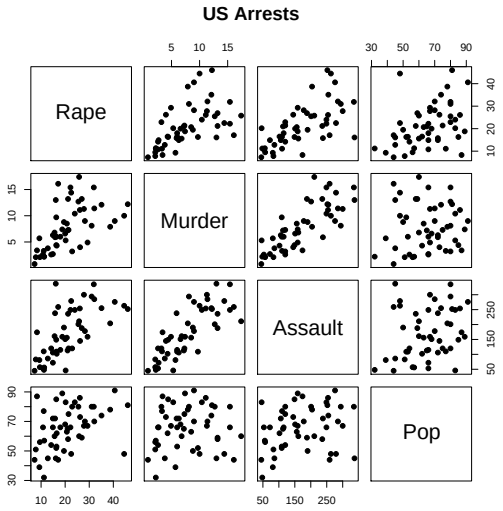


Figure 1.3: Pairwise scatter plots for three US arrest rates and percent of population living in Urban areas

additional library, and when you load the library you need to tell R where to find it. It is just possible to envisage the three dimensions on the printed page.

```
> install.packages("scatterplot3d", lib = "u:/STAT3401/mvm")
> library(scatterplot3d, lib.loc = "u:/STAT3401/mvm/")
> data(trees)
>  s3d <- scatterplot3d(USArrests[,-3], type="h", highlight.3d=TRUE,
      angle=55, scale.y=0.7, pch=16, main = "USArrests")
```

1.4.4 Other methods

Other useful methods will be considered in the lab-session, such as glyphs and arrows. For example there is a rather simple glyph plot available here:

```
glyphs(Assault, Murder, Rape, UrbanPop)
```

The idea of the glyph is that two of the variables are represented as the x and y co-ordinates (as usual), but a futher variable can be represented by the angle of the arrow, and a fourth variable as the length of the arrow.

Interactive graphics offer these facilities, and many more. There are programs such as GGobi (`www.ggobi.org`) which allow extensive multivariate investigation such as linked / brushed plots and "grand tours".

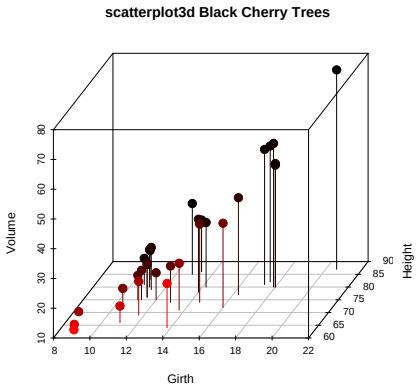


Figure 1.4: 3d scatterplot of US arrests

Profiles

Just as much ingenuity was extended before modern colour systems. Another approach is Andrews Curves, described as a function between $-\pi < t < \pi$

$$f_x(t) = x_1/\sqrt{2} + x_2 \sin 2 + x_3 \cos t + x_4 \sin 2t + x_5 \cos 2t + \ldots$$

You may like to consider at some stage (perhaps not today) how you could write an R function that plots Andrew's curves? (there's a function in the `mvmmisc.R` file).

Try creating a matrix of data values from Fisher's Iris data, and a column of species names. Then call up the andrews curves function:

```
> iris.data <- iris[,-5]
> iris.species <- iris[,5]
> andrews.curves(iris.data, iris.species)
```

However, a simpler profile plots is available from the MASS library:

```
> library(MASS)
> parcoord(USArrests)
```

The idea is that not only are the values of each individual variable represented, but also the patterns of different individuals can be seen.

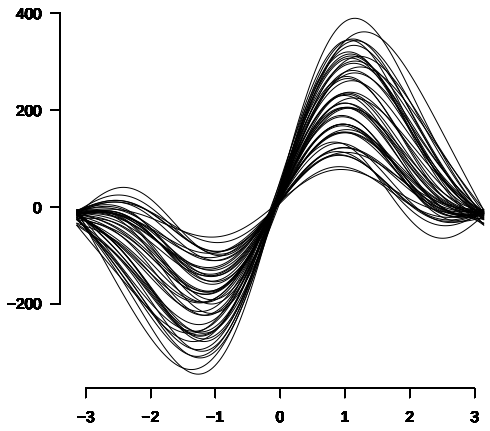


Figure 1.5: Andrews Curves of US arrests data

If you now try looking at Fisher's Iris data (the [,-5] drops the species column which is a factor and cannot be plotted)

```
> parcoord(iris[,-5])
```

You can also tell `parcoord()` to colour the profiles in according to the species.

```
> parcoord(iris[,-5], col = as.numeric(iris[,5]))
```

1.5 Animated exploration

This shows an rgl with ellipsoids.

You can use the help system to find more information on the datasets (e.g. type `> ?longley`).

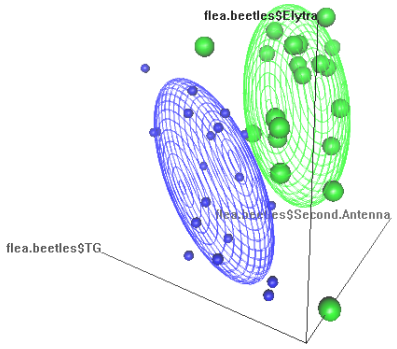


Figure 1.6: rgl animated view of first three variables of flea beetle data

Chapter 2

Matrix manipulation

It is convenient to represent multivariate data by means of $n \times p$ matrix such as $\boldsymbol{X}$. We could consider the `USArrests` data in this way. We follow the convention of using n to denote the number of rows of individuals who have been observed, and p to denote the number of columns (variables). We will formalise some some aspects from linear algebra that will be important in understanding multivariate analysis. These are very brief notes, there is a wealth of readable material on linear algebra as well as material specific for statistical applications such as Healy (2000) and Schott (1997). There is also an interesting presentation from a more geometric perspective in Wickens (1995) which supplements more algebraic presentations of matrix concepts.

2.1 Vectors

Consider a vector $\boldsymbol{x} \in \mathbb{R}^p$, by convention this is thought of as a column vector:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

A row vector such as $\begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}$ will be denoted by $\boldsymbol{x}^T$.

A vector is a basic unit of numbers within **R** , but the **R** objects don't entirely conform to a formal mathematical definition (look at the way vecctor recycling works for example) and some caution is needed. The following instruction:

```
> x <- c(3.289, 4.700, 10.400)
```

assigns the values to the object `x` creating the following **R** vector:

$$\boldsymbol{x} = \begin{pmatrix} 3.289 \\ 4.700 \\ 10.400 \end{pmatrix}$$

The default print method in **R** gives these in the most compact form:

```
> x
[1]  [1]  3.289  4.700 10.400
```

but forcing this into a matrix object with `as.matrix()` confirms its dimensionality:

```
> as.matrix(x)
       [,1]
[1,]  3.289
[2,]  4.700
[3,] 10.400
```

and taking the transpose of this vector using `t()` does produce a row vector as expected:

```
> t(x)
      [,1] [,2] [,3]
[1,] 3.289  4.7 10.4
```

2.1.1 Vector multiplication; the inner product

We first define the inner product of two vectors. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ this gives a scalar:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y} = \sum_{j=1}^{p} x_j y_j = \boldsymbol{y}^T \boldsymbol{x}$$

In other words, we find the product of corresponding elements of each vector (the product of the first element of the row vector and the first element of the column vector), and then find the sum of all these products:

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \underbrace{x_1y_1 + x_2y_2 + \dots + x_ny_n}_{\text{One number; the sum of all the individual products}}$$

To give a simple example, with $\boldsymbol{x}^T = (4, 1, 3, 2)$ and $\boldsymbol{y} = \begin{pmatrix} 1 \\ -1 \\ 3 \\ 0 \end{pmatrix}$ we have:

$$\begin{pmatrix} 4 & 1 & 3 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ -1 \\ 3 \\ 0 \end{pmatrix} = \underbrace{4 \times 1 + 1 \times (-1) + 3 \times 3 + 2 \times 0}_{} = 12$$

In **R** the inner product can be simply obtained using %*%, for example:

```
> x <- c(4, 1, 3, 2)
> y <- c(1, -1, 3, 0)
> t(x) %*% y
     [,1]
[1,]   12
```

which returns the answer as a scalar. Note that using * without the enclosing %% yields a vector of the same length of $\boldsymbol{x}$ and $\boldsymbol{y}$ where each element is the product of the corresponding elements of $\boldsymbol{x}$ and $\boldsymbol{y}$, and may do other unexpected things using vector recycling.

2.1.2 Outer product

Note that if $\boldsymbol{x}^T\boldsymbol{y}$ is the inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the *outer* product is given by $\boldsymbol{x}\boldsymbol{y}^T$. For vectors, it can be computed by `x %*% t(y)`; but as we will find later, outer product operations are defined for arrays of more than one dimension as `x %o% y` and `outer(x,y)`

2.1.3 Vector length

An important concept is the length of a vector, also known as the Euclidean norm or the modulus. It is based on a geometric idea and expresses the distance of a given vector from the origin:

$$|\boldsymbol{x}| = \langle \boldsymbol{x}, \boldsymbol{x} \rangle^{1/2} = \left(\sum_{j=1}^{p} x_j^2 \right)^{1/2}$$

A *normalised* vector is one scaled to have unit length, for the vector $\boldsymbol{x}$ this can be found by taking $\frac{1}{|\boldsymbol{x}|}\boldsymbol{x}$ which is trivial in **R** :

```
> z <- x / sqrt(t(x) %*% x)
> z
[1] 0.7302967 0.1825742 0.5477226 0.3651484
> t(z) %*% z ## check the length of the normalised vector
     [,1]
[1,]    1
```

2.1.4 Orthogonality

Two vectors **x** and **y**, of order $k \times 1$ are orthogonal if $\boldsymbol{xy} = 0$. Furthermore, if two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal *and* of unit length, i.e. if $\boldsymbol{xy} = 0$, $\boldsymbol{x}^T\boldsymbol{x} = 1$ and $\boldsymbol{y}^T\boldsymbol{y} = 1$ then they are orthonormal.

More formally, a set $\{\boldsymbol{e}_i\}$ of vectors in $\mathbb{R}^p$ is orthonormal if

$$\boldsymbol{e}_i^T\boldsymbol{e}_j = \delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

Where δ_{ij} is referred to as the Kronecker delta.

2.1.5 Cauchy-Schwartz Inequality

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq |\boldsymbol{x}|\ |\boldsymbol{y}|, \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$$

with equality if and only if $\boldsymbol{x} = \lambda\boldsymbol{y}$ for some $\lambda \in \mathbb{R}$. Proof of this inequality is given in many multivariate textbooks such as Bilodeau and Brenner (1999). We won't use this result itself, but will actually consider the extended Cauchy-Scwartz inequality later.

2.1.6 Angle between vectors

The cosine of the angle between two vectors is given by:

$$\cos(\theta) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{|\boldsymbol{x}|\ |\boldsymbol{y}|}$$

It can be conveniently calculated in **R** :

```
> cor(x,y)
```

2.2 Matrices

We now consider some basic properties of matrices, and consider some basic operations on them that will become essential as we progress. Consider the data matrix $\boldsymbol{X}$, containing the `USArrests` data, a 50×4 matrix, i.e. with $n = 50$ rows refering to States and $p = 4$ columns refering to the variables measuring different arrest rates. To indicate the order of this matrix it could be described fully as $\boldsymbol{X}_{50,4}$; this convention is followed in **R** as a call to `dim(USArrests)` will confirm. Each element in this matrix can be denoted by x_{ij} where i denotes the particular row (here state) and j the particular column (here arrest rate). Hence $x_{6\ 3} = 38.7$.

In order to create a matrix in **R** the dimension has to be specified in the call to `matrix()`. It should be very carefully noted that the default is to fill a matrix by columns, as indicated here:

```
> mydata <- c(1,2,3,4,5,6)
> A <- matrix(mydata, 3,2)
> A
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

If this is not convenient, **R** can be persuaded to fill matrices by rows rather than by columns by including the argument `byrow = TRUE` in the call to `matrix`. It is also possible to coerce other objects (such as data frames) to a matrix using `as.matrix()` and `data.matrix()`; the former producing a character matrix if there are any non-numeric variables present, the latter coercing everything to a numeric format.

2.2.1 Transposing matrices

Transposing matrices simply involves turning the first column into the first row. A transposed matrix is denoted by a superscripted T, in other words $\mathbf{A^T}$ is the transpose of $\mathbf{A}$.

$$If\ \mathbf{A} = \begin{pmatrix} 3 & 1 \\ 5 & 6 \\ 4 & 4 \end{pmatrix} \text{ then } \mathbf{A^T} = \begin{pmatrix} 3 & 5 & 4 \\ 1 & 6 & 4 \end{pmatrix}$$

As with vectors, transposing matrices in **R** simply requires a call to `t()`, the dimensions can be checked with `dim()`.

```
> Atrans <- t(A)
> Atrans
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> dim(Atrans)
[1] 2 3
```

2.2.2 Some special matrices

Symmetric matrices

We mention a few "special" matrix forms that will be encountered. We firstly note that *symmetric* matrices are symmetric around the diagonal $i = j$. For matrix $\boldsymbol{A}$, it is symmetric whenever $a_{ij} = a_{ji}$. The correlation matrix and the variance-covariance matrix are the most common symmetric matrices we will encounter, we will look at them in more detail later, for now note that we can obtain the (symmetric) correlation matrix as follows:

```
> cor(USArrests)
             Murder   Assault   UrbanPop      Rape
Murder   1.00000000 0.8018733 0.06957262 0.5635788
Assault  0.80187331 1.0000000 0.25887170 0.6652412
UrbanPop 0.06957262 0.2588717 1.00000000 0.4113412
Rape     0.56357883 0.6652412 0.41134124 1.0000000
```

Diagonal Matrices

Given it's name, it is perhaps obvious that a diagonal matrix has elements on the diagonal (where $i = j$) and zero elsewhere (where $i \neq j$). For example, the matrix $\boldsymbol{A}$ given as follows:

$$\mathbf{A} = \begin{pmatrix} 13 & 0 & 0 \\ 0 & 27 & 0 \\ 0 & 0 & 16 \end{pmatrix}$$

is a diagonal matrix. To save paper and ink, $\boldsymbol{A}$ can also be written as:

$$\mathbf{A} = diag \begin{pmatrix} 13 & 27 & 16 \end{pmatrix}$$

It is worth noting that the `diag()` command in **R** , as shown below, lets you both *overwrite* the diagonal elements of matrix and *extract* the diagonal elements depending how it is used:

```
> mydataD <- c(13, 27, 16)
> B <- diag(mydataD)
> B
     [,1] [,2] [,3]
[1,]   13    0    0
[2,]    0   27    0
[3,]    0    0   16
> diag(B)
[1] 13 27 16
```

It is also worth noting that when "overwriting", the size of the matrix to be over-written can be inferred from the dimensionality of diagonal.

Identity Matrix

One special diagonal matrix is the identity matrix, which has a value of 1 at each position on the diagonal and 0 elsewhere. Here, all we need to know is the size. So $\mathbf{I_4}$ tells us that we have the following matrix:

$$\mathbf{I_4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This can be created in a variety of ways in **R** , such as `I4 <- diag(rep(1,4))`

Ones

We also need to define a vector of ones; $\mathbf{1}_p$, a $p \times 1$ matrix containing only the value 1. There is no inbuilt function in **R** to create this vector, it is easily added:

```
> ones <- function(p){
  Ones <- matrix(1,p,1)
  return(Ones)
}
```

Zero matrix

Finally, $\mathbf{0}$ denotes the zero matrix, a matrix of zeros. Unlike the previously mentioned matrices this matrix can be any shape you want. So, for example:

$$\mathbf{0}_{2\ 3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

2.2.3 Equality and addition

A little more care is needed in defining basic mathematical operations on matrices. Considering the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we consider their equality $\boldsymbol{A} = \boldsymbol{B}$ if any only if:

- $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same size, and
- the ijth element of $\boldsymbol{A}$ is equal to the ijth element of $\boldsymbol{A}$ for all $1 \leq i \leq r$ and $1 \leq j \leq n$

A consequence of this is that the following two matrices are equal:

$$\begin{bmatrix} 138.8149 & 187.52 & 394.86 \\ 187.5200 & 267.00 & 559.00 \\ 394.8600 & 559.00 & 1200.00 \end{bmatrix} = \begin{bmatrix} 138.8149 & 187.52 & 394.86 \\ 187.5200 & 267.00 & 559.00 \\ 394.8600 & 559.00 & 1200.00 \end{bmatrix}$$

(which seems like an obvious and fussy thing to say) *but* the following two zero matrices are not equal:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \neq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Adding and subtracting are fairly straightforward. Provided $\boldsymbol{A}$ and $\boldsymbol{A}$ have the same size, $\boldsymbol{A} + \boldsymbol{B}$ and $\boldsymbol{A} - \boldsymbol{B}$ are defined by each of these operations being carried out on individual elements of the matrix. For example:

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} + \begin{pmatrix} 0 & 2 & 3 \\ -1 & -2 & -3 \end{pmatrix} = \begin{pmatrix} 1+0 & 3+2 & 5+3 \\ 2+-1 & 4+-2 & 6+-3 \end{pmatrix} = \begin{pmatrix} 1 & 5 & 8 \\ 1 & 2 & 3 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} - \begin{pmatrix} 0 & 2 & 3 \\ -1 & -2 & -3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 \\ 3 & 6 & 9 \end{pmatrix}$$

Addition and subtraction are straightforward enough in **R** :

```
> A <- matrix(c(1,2,3,4,5,6),2,3)
> A
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> B <- matrix(c(0,-1,2,-2,3,-3),2,3)
> B
     [,1] [,2] [,3]
[1,]    0    2    3
[2,]   -1   -2   -3
> A + B
     [,1] [,2] [,3]
[1,]    1    5    8
[2,]    1    2    3
> A - B
     [,1] [,2] [,3]
[1,]    1    1    2
[2,]    3    6    9
```

Matrix addition follows all the normal arithmetic rules, i.e.

$$\begin{aligned} \text{Commutative law} \quad & \mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \\ \text{Associative law} \quad & \mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C} \end{aligned}$$

Matrix multiplication however follows vector multiplication and therefore does not follow the same rules as basic multiplication.

2.2.4 Multiplication

A scalar is a matrix with just one row and one column, i.e. a single number. In other words, 0.4 could be a scalar or a 1×1 matrix. It's worth re-capping that multiplication by a scalar is easy enough, we just multiply every element in the matrix by the scalar.

So if $\mathbf{k} = 0.4$, and

$$\mathbf{A} = \begin{pmatrix} 1 & 5 & 8 \\ 1 & 2 & 3 \end{pmatrix}$$

we can calculate $\mathbf{kA}$ as:

$$\mathbf{kA} = 0.4 \times \begin{pmatrix} 1 & 5 & 8 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 0.4 & 2 & 3.2 \\ 0.4 & 0.8 & 1.6 \end{pmatrix}$$

When multiplying two matrices, it should be noted first that they must be conformable. The number of columns in the first matrix must match the number of rows in the second. As matrix multiplication has been defined, the result will be a matrix with as many rows as the first matrix and as many columns as the second. For example, with our vectors above in section 2.1.1 , we had $\boldsymbol{A_{1\ 4}} \times \boldsymbol{B_{4\ 1}} = \boldsymbol{C_{1\ 1}}$. More generally multiplication proceeds with matrix size as follows: $\boldsymbol{A_{m\ n}} \times \boldsymbol{B_{n\ p}} = \boldsymbol{C_{m\ p}}$.

It *may* help to think about the vector operations and extend them to matrices. There are other ways of thinking about matrix multiplication, most multivariate text books have an appendix on matrix algebra and there are vast tomes available covering introductory linear algebra. However, one explanation of matrix multiplication is given here. We want to find $\boldsymbol{A} \times \boldsymbol{B}$ where

$$\mathbf{A} = \begin{pmatrix} 1 & 5 \\ 1 & 2 \\ 3 & 8 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} 1 & 4 \\ 3 & 2 \end{pmatrix}$$

If **A** is of size $m \times n$ it could be considered as consisting of a row of vectors $\boldsymbol{a_1^T}, \boldsymbol{a_1^T}, \ldots, \boldsymbol{a_m^T}$, which in this case corresponds to $\boldsymbol{a_1^T} = (1, 5), \boldsymbol{a_2^T} = (1, 2)$ and $\boldsymbol{a_3^T} = (3, 8)$. Likewise, we can consider $\mathbf{B}$ consisting of $\boldsymbol{b_1} = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$ and $\boldsymbol{b_1} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$. In other words, we are trying to multiply together:

$$\mathbf{A} = \begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} b_1 & b_2 \end{pmatrix}$$

We can define the multiplication operation for matrices generally as:

$$\mathbf{AB} = \begin{pmatrix} a_1^T \\ a_2^T \\ \ldots \\ a_m^T \end{pmatrix} \begin{pmatrix} b_1 & b_2 & \ldots & b_p \end{pmatrix} = \begin{pmatrix} a_1^T b_1 & a_1^T b_2 & \ldots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \ldots & a_1^T b_p \\ \vdots & \vdots & & \vdots \\ a_3^T b_1 & a_3^T b_2 & \ldots & a_m^T b_p \end{pmatrix}$$

In other words, we need to multiply row i of $\boldsymbol{A}$ by column j of $\boldsymbol{B}$ to give element ij of the result. For example, note that $\boldsymbol{a_1^T b_1} = \begin{pmatrix} 1 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 4 \end{pmatrix} = 1 \times 1 + 5 \times 3 = 16$. Carrying out this operation on our matrices above gives:

$$\mathbf{AB} = \begin{pmatrix} 1 & 5 \\ 1 & 2 \\ 3 & 8 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 3 & 2 \end{pmatrix} = \begin{pmatrix} 16 & 14 \\ 7 & 8 \\ 27 & 28 \end{pmatrix}$$

In **R** , we only need to use the %*% operator to ensure we are getting matrix multiplication:

```
> A <- matrix(c(1,1,3,5,2,8),3,2)
> A
     [,1] [,2]
[1,]    1    5
[2,]    1    2
[3,]    3    8
> B <- matrix(c(1,3,4,2),2,2)
> B
     [,1] [,2]
[1,]    1    4
[2,]    3    2
> A %*% B
     [,1] [,2]
[1,]   16   14
[2,]    7    8
[3,]   27   28
```

Note that you can't multiply non-conformable matrices; this is one place in **R** where you get a clearly informative error message:

```
> B %*% A
Error in B %*% A : non-conformable arguments
```

It is particularly important to use the correct *matrix multiplication* argument. Depending on the matrices you are working with (if they both have the same dimensions), using the usual $*$ multiplication operator will give you the Hadamard product, the element by element product of the two matrices which is rarely what you want:

```
> C <- matrix(c(1,1,3,5),2,2)
> C %*% B ## correct call for matrix multiplication
     [,1] [,2]
[1,]   10   10
[2,]   16   14
> C * B ## Hadamard Product!!!
     [,1] [,2]
[1,]    1   12
[2,]    3   10
```

We saw earlier that matrix addition was commutative and associative. But as you can imagine, given the need for comformability some differences may be anticipated between conventional multiplication and matrix multiplication. Generally speaking, matrix multiplication is not commutative (you may like to think of exceptions):

$$\begin{aligned} \text{(non-commutative)} \quad & \mathbf{A} \times \mathbf{B} \neq \mathbf{B} \times \mathbf{A} \\ \text{Associative law} \quad & \mathbf{A} \times (\mathbf{B} \times \mathbf{C}) = (\mathbf{A} \times \mathbf{B}) \times \mathbf{C} \end{aligned}$$

And the distributive laws of multiplication over addition apply as much to matrix as conventional multiplication:

$$\begin{aligned} \mathbf{A} \times (\mathbf{B} + \mathbf{C}) &= (\mathbf{A} \times \mathbf{B}) + (\mathbf{A} \times \mathbf{C}) \\ (\mathbf{A} + \mathbf{B}) \times \mathbf{C} &= (\mathbf{A} \times \mathbf{C}) + (\mathbf{B} \times \mathbf{C}) \end{aligned}$$

But there are a few pitfalls if we start working with transposes. Whilst

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$$

note that:

$$(\mathbf{A} \times \mathbf{B})^T = \mathbf{B}^T \times \mathbf{A}^T$$

Trace of a matrix

The trace of a matrix is the quite simply the sum of its diagonal elements. This is an interesting concept in many ways, but it turns out in one specific context, when applied to the covariance matrix, this has an interpretation as the total sample variance. There is no inbuilt function in **R** to calculate this value, you need to use `sum(diag(X))`

Note that if you have two conformable matrices $\boldsymbol{A}$ e.g. $\begin{pmatrix} 2 & 5 \\ 0 & 7 \\ 4 & 3 \end{pmatrix}$ and $\boldsymbol{B}$ e.g. $\begin{pmatrix} 4 & 2 & 1 \\ 6 & 3 & 2 \end{pmatrix}$, $trace(\boldsymbol{AB}) = trace(\boldsymbol{BA})$

2.3 Crossproduct matrix

Given the data matrix $\boldsymbol{X}$, the crossproduct, sometimes more fully referred to as the "sum of squares and crossproducts" matrix is given by $\boldsymbol{X}^T\boldsymbol{X}$. The diagonals of this matrix are clearly the sum of squares of each column. Whilst this can be computed in **R** using `X %*% t(X)` there are some computational advantages in using the dedicated function `crossprod(X)` For example, coercing the `USArrests` data to a matrix we can obtain the sum of squares and crossproducts matrix for these

data as follows:

```
B <- crossprod(as.matrix(USArrests))
```

So if $\boldsymbol{X}$ is the `USArrests` data,

$$\boldsymbol{X^T X} = \begin{bmatrix} 3962.20 & 80756.00 & 25736.20 & 9394.32 \\ 80756.00 & 1798262.00 & 574882.00 & 206723.00 \\ 25736.20 & 574882.00 & 225041.00 & 72309.90 \\ 9394.32 & 206723.00 & 72309.90 & 26838.62 \end{bmatrix}$$

If we define some sample estimators as follows:

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i = \frac{1}{n}\boldsymbol{X}^T\mathbf{1} \tag{2.1}$$

So for example we can find the sample mean for the USArrests data as:

```
> n <- dim(USArrests)[1] ## extract n; here 50
> one <- ones(n)
> 1/n * t(USArrests) %*% one
> mean(USArrests) ## check results against in-built function
```

We can use matrix algebra to obtain an unbiased estimate of the sample covariance matrix $\boldsymbol{S}$ as follows:

$$\begin{aligned} \boldsymbol{S} &= \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \\ &= \sum_{i=1}^{n}\boldsymbol{x}_i\sum_{i=1}^{n}\boldsymbol{x}_i^T - \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T \\ &= \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} - \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T \\ &= \frac{1}{n-1}\left(\boldsymbol{X}^T\boldsymbol{X} - \frac{1}{n}\boldsymbol{X}^T\mathbf{1}\mathbf{1}^T\boldsymbol{X}\right) \end{aligned}$$

From this, we can define the centering matrix $\boldsymbol{H}$:

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$$

and so arrive at an alternative expression for $\boldsymbol{S}$ using this centering matrix:

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{H}\boldsymbol{X} \tag{2.2}$$

Idempotent matrices

It may be noted that $\boldsymbol{H}$ is idempotent, i.e. $\boldsymbol{H} = \boldsymbol{H}^T$ and $\boldsymbol{H} = \boldsymbol{H}^2$.

In calculating $\boldsymbol{H}$ in R it might be clearer to set the steps out in a function:

```
centering <- function(n){
 I.mat <- diag(rep(1, n))
 Right.mat <- 1/n * ones(n) %*% t(ones(n))
 H.mat <- I.mat - Right.mat
 return(H.mat)
}
```

And our matrix method for finding an estimate of the sample covariance using this centering procedure can also be set out in a function:

```
S.mat <- function(X, H){
n <- dim(X)[1] ## number of rows
H.mat <- centering(n)
S <- 1/(n-1) * t(X) %*% H.mat %*% X
return(S)
}
```

So, to estimate the sample covariance with this function we need to make sure our data are in the form of matrix. We also compare the results with the inbuilt function `cov()`:

```
X <- as.matrix(USArrests)
S.mat(X)
cov(USArrests)
```

It may be worth clarifying the information contained in the matrix we have just obtained. The covariance matrix (more fully referred to as the variance-covariance matrix) contains information on the variance of each of the variables as well as information on pairwise covariance. We will formalise our understanding of estimators later, but for now note that it could be considered as an estimate of:

$$\boldsymbol{\Sigma} = \boldsymbol{V}\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \begin{pmatrix} var(X_1) & cov(X_1, X_2) & cov(X_1, X_3) & cov(X_1, X_4) \\ cov(X_2, X_1) & var(X_2) & cov(X_2, X_3) & cov(X_2, X_4) \\ cov(X_3, X_1) & cov(X_3, X_2) & var(X_3) & cov(X_3, X_4) \\ cov(X_4, X_1) & cov(X_4, X_2) & cov(X_4, X_3) & var(X_4) \end{pmatrix}$$

For the US Arrests data, as we have seen:

$$\boldsymbol{S} = \begin{pmatrix} 18.97 & 291.06 & 4.39 & 22.99 \\ 291.06 & 6945.17 & 312.28 & 519.27 \\ 4.39 & 312.28 & 209.52 & 55.77 \\ 22.99 & 519.27 & 55.77 & 87.73 \end{pmatrix}$$

2.3.1 Powers of matrices

We set out some definitions of matrix powers as they will come in useful later.For all matrices, we define $\mathbf{A^0} = \mathbf{I}$, the identity matrix and $\mathbf{A^1} = \mathbf{A}$. We will next define $\mathbf{A^2} = \mathbf{AA}$ (if you think about it a bit you could see that $\mathbf{A}$ must be a square matrix, otherwise we couldn't carry out this multiplication). Using these definitions for matrix powers means that all the normal power arithmetic applies. For example, $\mathbf{A^m} \times \mathbf{A^n} = \mathbf{A^n} \times \mathbf{A^m} = \mathbf{A^{m+n}}$. If you look closely, you can also see that the powers of a matrix are commutative which means that we can do fairly standard algebraic factorisation. For example:

$$\mathbf{I} - \mathbf{A^2} = (\mathbf{I} + \mathbf{A})(\mathbf{I} - \mathbf{A})$$

which is a result we can use later.

2.3.2 Determinants

The determinant of a *square* $p \times p$ matrix **A** is denoted as $|\boldsymbol{A}|$. Finding the determinant of a 2×2 matrix is easy:

$$|\boldsymbol{A}| = det \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

For matrices of order > 2, partitioning the matrix into "minors" and "cofactors" is necessary. Consider the following 3×3 matrix.

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{13} \end{pmatrix}$$

Any element a_{ij} of this matrix has a corresponding square matrix formed by eliminating the row (i) and column (j) containing a_{ij}. So if we were considering a_{11}, we would be interested in the square matrix $\boldsymbol{A_{-11}} = \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{13} \end{pmatrix}$. The determinant of this reduced matrix, $|\boldsymbol{A_{-11}}|$ is called the minor

of a_{11}, and the product $c_{ij} = (-1)^{i+j}|A_{-ij}| = -1^{1+1}|A_{-11}| = |A_{11}|$ is called the cofactor of a_{11}. The determinant of $\boldsymbol{A}$ can be expressed as the sum of minors and cofactors of any row or column of $\boldsymbol{A}$.

Thus:

$$|\boldsymbol{A}| = \Sigma_{j=1}^{p} a_{ij} c_{ij}$$

and as can be seen, this can get terribly recursive if you're working by hand! Working an example through:

$$\text{If} \boldsymbol{A} = \begin{pmatrix} 3 & 4 & 6 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{pmatrix}$$

Then $|\boldsymbol{A}| = a_{i1}c_{i1} + a_{i2}c_{i2} + a_{i3}c_{i3}$. If $i = 1$ then:

$$\begin{aligned} c_{11} &= (-1)^{1+1} \begin{vmatrix} 2 & 3 \\ 7 & 9 \end{vmatrix} = (18 - 21) = -3 \\ c_{11} &= (-1)^{1+2} \begin{vmatrix} 1 & 3 \\ 5 & 9 \end{vmatrix} = -(9 - 15) = 6 \\ c_{11} &= (-1)^{1+1} \begin{vmatrix} 1 & 2 \\ 5 & 7 \end{vmatrix} = (7 - 10) = -3 \end{aligned}$$

So $|\boldsymbol{A}| = 3(-3) + 4(6) + 6(-3) = -3$.

In **R** , `det()` tries to find the determinant of a matrix.

```
> D <- matrix(c(5,3,9,6),2,2)
> D
     [,1] [,2]
[1,]    5    9
[2,]    3    6
> det(D)
[1] 3
> E <- matrix(c(1,2,3,6),2,2)
> E
     [,1] [,2]
[1,]    1    3
[2,]    2    6
> det(E)
[1] 0
```

Some useful properties of determinants:

- The determinant of a diagonal matrix (or a triangular matrix for that matter) is the product of the diagonal elements. (Why?).
- For any scalar k, $|k\boldsymbol{A}| = k^n|\boldsymbol{A}|$, where $\boldsymbol{A}$ has size $n \times n$.
- If two rows or columns of a matrix are interchanged, the sign of the determinant changes.
- If two rows or columns are equal or proportional (see material on rank later), the determinant is zero.
- The determinant is unchanged by adding a multiple of some column (row) to any other column (row).
- If all the elements or a column / row are zero then the determinant is zero.
- If two $n \times n$ matrices are denoted by $\boldsymbol{A}$ and $\boldsymbol{B}$, then $|\boldsymbol{AB}| = |\boldsymbol{A}|.|\boldsymbol{B}|$.

The determinant of a variance-covariance has a rather challenging interpretation as the generalised variance.

2.3.3 Rank of a matrix

Rank denotes the number of linearly independent rows or columns. For example:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 5 & -1 \\ 0 & 1 & -1 \end{pmatrix}$$

This matrix has dimension 3×3, but only has rank 2. The second column $\boldsymbol{a_2}$ can be found from the other two columns as $\boldsymbol{a_2} = 2\boldsymbol{a_1} - \boldsymbol{a_3}$.

If all the rows and columns of a square matrix **A** are linearly independent it is said to be of full rank and non-singular.

If **A** is singular, then $|\boldsymbol{A}| = 0$.

2.4 Matrix inversion

If **A** is a non-singular $p \times p$ matrix, then there is a unique matrix **B** such that $\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}$, where **I** is the identity matrix given earlier. In this case, **B** is the inverse of **A**, and denoted $\boldsymbol{A^{-1}}$.

Inversion is quite straightforward for a 2×2 matrix.

$$\text{If } \boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ then } \boldsymbol{A}^{-1} = \frac{1}{|\boldsymbol{A}|} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

More generally for a matrix of order $n \times n$, the (j,k)th entry of $\boldsymbol{A^{-1}}$ is given by:

$$\left[\frac{|\boldsymbol{A}_{-\boldsymbol{jk}}|}{|\boldsymbol{A}|}\right]^{(-1)^{j+k}},$$

where $\boldsymbol{A}_{-\boldsymbol{jk}}$ is the matrix formed by deleting the jth row and kth column of $\boldsymbol{A}$. Note that a singular matrix has no inverse since its determinant is 0.

In **R** , we use `solve()` to invert a matrix (or solve a system of equations if you have a second matrix in the function call, if we don't specify a second matrix R assumes we want to solve against the identity matrix, which mean finding the inverse).

```
> D <- matrix(c(5,3,9,6),2,2)
> solve(D)
     [,1]      [,2]
[1,]    2 -3.000000
[2,]   -1  1.666667
```

Some properties of inverses:

- The inverse of a symmetric matrix is also symmetric.
- The inverse of the transpose of $\boldsymbol{A}$ is the transpose of $\boldsymbol{A}^{-1}$.
- The inverse of the product of several square matrices is a little more subtle: $(\boldsymbol{ABC})^{-1} = \boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. If c is a non-zero scalar then $(c\boldsymbol{A})^{-1} = c^{-1}\boldsymbol{A}^{-1}$.
- The inverse of a diagonal matrix is really easy - the reciprocals of the original elements.

2.5 Eigen values and eigen vectors

These decompositions will form the core of at least half our multivariate methods (although we need to mention at some point that we actually tend to use the singular value decomposition as a means of getting to these values). If $\boldsymbol{A}$ is a square $p \times p$ matrix, the eigenvalues (latent roots, characteristic roots) are the roots of the equation:

$$|\boldsymbol{A} - \lambda \boldsymbol{I}| = \boldsymbol{0}$$

This (characteristic) equation is a polynomial of degree p in λ. The roots, the eigenvalues of $\boldsymbol{A}$ are denoted by $\lambda_1, \lambda_2, \ldots, \lambda_p$. For each eigen value λ_i there is a corresponding eigen vector $\boldsymbol{e}_i$ which can be found by solving:

$$(\boldsymbol{A} - \lambda_i \boldsymbol{I})\boldsymbol{e_i} = \boldsymbol{0}$$

There are many solutions for $\boldsymbol{e_i}$. For our (statistical) purposes, we usually set it to have length 1, i.e. we obtain a normalised eigenvector for λ_i by $\boldsymbol{a_i} = \frac{\boldsymbol{e_i}}{\sqrt{\boldsymbol{e_i}^T \boldsymbol{e_i}}}$

We pause to mention a couple of results that will be explored in much more detail later:

(a) $trace(\boldsymbol{A}) = \Sigma_{i=1}^{p} \lambda_i$

(b) $|\boldsymbol{A}| = \prod_{i=1}^{p} \lambda_i$

Also, if $\boldsymbol{A}$ is symmetric:

(c) The normalised eigenvectors corresponding to unequal eigenvalues are orthonormal (this is a bit of circular definition, if the eigenvalues are equal the corresponding eigenvectors are not unique, and one "fix" is to choose orthonormal eigenvectors).

(d) Correlation and covariance matrices: are symmetric positive definite (or semi-definite). If such a matrix is of full rank p then all the eigen values are positive. If the matrix is of rank $m < p$ then there will be m positive eigenvalues and $p - m$ zero eigenvalues.

We will look at the `eigen()` function in **R** to carry out these decompositions later.

2.6 Singular Value Decomposition

To be added.

2.7 Extended Cauchy-Schwarz Inequality

We met the rather amazing Cauchy Schwartz inequality earlier in section 2.1.5. Beautiful as this result may be, we actually need to use the *extended* Cauchy Schwartz inequality. For any non-zero vectors $\boldsymbol{x} \in \mathbb{R}$ and $\boldsymbol{y} \in \mathbb{R}$, with any positive definite $p \times p$ matrix $\boldsymbol{S}$:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \leq (\boldsymbol{x}^T \boldsymbol{S} \boldsymbol{x})(\boldsymbol{y}^T \boldsymbol{S}^{-1} \boldsymbol{y}), \text{ for all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$$

with equality if and only if $\boldsymbol{x} = \lambda \boldsymbol{S} \boldsymbol{y}$ for some $\lambda \in \mathbb{R}$. Proofs are available for this result (Flury, 1997, page 291). We will use this result when developing methods for discriminant analysis.

2.8 Partitioning

Finally, note that we can partition a large matrix into smaller ones:

$$\left(\begin{array}{cc|c} 2 & 5 & 4 \\ \hline 0 & 7 & 8 \\ 4 & 3 & 4 \end{array}\right)$$

So we could work with submatrices such as $\begin{pmatrix} 0 & 7 \\ 4 & 3 \end{pmatrix}$.

e.g. If $\boldsymbol{X}$ was partitioned as $\begin{pmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{pmatrix}$ and $\begin{pmatrix} \boldsymbol{Y_1} & \boldsymbol{Y_2} & \boldsymbol{Y_3} \end{pmatrix}$ then:

$$\boldsymbol{XY} = \begin{pmatrix} \boldsymbol{X_1 Y_1} & \boldsymbol{X_1 Y_2} & \boldsymbol{X_1 Y_3} \\ \boldsymbol{X_2 Y_1} & \boldsymbol{X_2 Y_2} & \boldsymbol{X_2 Y_3} \end{pmatrix}$$

2.9 Exercises

1. Which of the following are orthogonal to each other:

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ -2 \\ 3 \\ -4 \end{pmatrix} \boldsymbol{y} = \begin{pmatrix} 6 \\ 7 \\ 1 \\ -2 \end{pmatrix} \boldsymbol{z} = \begin{pmatrix} 5 \\ -4 \\ 5 \\ 7 \end{pmatrix}$$

   Normalise each of the two orthogonal vectors.

2. Find vectors which are orthogonal to:

$$\boldsymbol{u} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \boldsymbol{v} = \begin{pmatrix} 2 \\ 4 \\ -1 \\ 2 \end{pmatrix}$$

3. Find vectors which are orthonormal to:

$$\boldsymbol{x} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \boldsymbol{y} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{5}{6} \end{pmatrix}$$

4. What are the determinants of:

$$(a) \begin{pmatrix} 1 & 3 \\ 6 & 4 \end{pmatrix} (b) \begin{pmatrix} 3 & 1 & 6 \\ 7 & 4 & 5 \\ 2 & -7 & 1 \end{pmatrix}$$

5. Invert the following matrices:

$$(a) \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 9 \end{pmatrix} (b) \begin{pmatrix} 2 & 3 \\ 1 & 5 \end{pmatrix} (c) \begin{pmatrix} 3 & 2 & -1 \\ 1 & 4 & 7 \\ 0 & 4 & 2 \end{pmatrix} (d) \begin{pmatrix} 1 & 1 & 1 \\ 2 & 5 & -1 \\ 3 & 1 & -1 \end{pmatrix}$$

6. Find eigenvalues and corresponding eigen vectors for the following matrices:

$$\boldsymbol{a} = \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix} \boldsymbol{b} = \begin{pmatrix} 1 & 2 \\ 3 & 2 \end{pmatrix} \boldsymbol{c} = \begin{pmatrix} 2 & -2 \\ -2 & 5 \end{pmatrix} \boldsymbol{d} = \begin{pmatrix} 2 & 2 \\ 2 & 5 \end{pmatrix}$$

$$\boldsymbol{e} = \begin{pmatrix} 1 & 4 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{f} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{g} = \begin{pmatrix} 13 & -4 & 2 \\ -4 & 13 & -2 \\ 2 & -2 & 10 \end{pmatrix}$$

7. Convert the following covariance matrix (you've seen it earlier) to a correlation matrix, calculate the eigenvalues and eigenvectors and verify that the eigen vectors are orthogonal.

$$\boldsymbol{g} = \begin{pmatrix} 13 & -4 & 2 \\ -4 & 13 & -2 \\ 2 & -2 & 10 \end{pmatrix}$$

Chapter 3

Measures of distance

We take a little time out here to consider some ideas regarding multivariate distance and introduce some properties of multivariate distance matrices. These concepts are most obviously relevant when considering multivariate technique such as cluster analysis and scaling methods, where we wish to examine the difference between individuals. In doing this, we need to find some definition of the concept of "difference between individuals", and will therefore consider a range of proximity measures. We also provide some discussion of difference between variables. These are currently important concepts in bio-informatic applications but earlier work in multivariate statistics involved consideration of variables which may be carrying similar information where cluster analysis of variables could be used as a preliminary data analytical exercise. We start by considering one particular measure, the Mahalanobis distance.

3.1 Mahalanobis Distance

The Mahalanobis distance has an important role in multivariate theory, albeit this is often an implied consideration rather than an explicit one. For example, development of forms of discriminant analysis considered in chapter 8 involve this measure. There are however a number of important distributional properties of the Mahalanobis distance which could be more used in determining multivariate normality. It should be noted that use of standard distance requires a parametric view of the world, and in particular it is most applicable for symmetric distributions. We follow Flury (1997) in providing the following exposition of the standard distance.

Firstly, if we consider the *univariate* standard distance we see that this is a measure of the absolute distance between two observations in units of their standard deviation. Given X, a random variable with mean μ and variance $\sigma^2 > 0$, the *standard distance*, between two numbers x_1 and x_2 is diven

by:

$$d(x_1, x_2) = \frac{|x_1 - x_2|}{\sigma}$$

Where $\sigma = 1$, we will find that this standard distance is the same as the Euclidean distance given later in section 3.3.1. The standardisation implied in this measure is important, for example, as will be noted later it is invariant under non-degenerate linear transformations. A univariate example would be given by considering $Y = \alpha X + b$, where $\alpha \neq 0$ and β are fixed constants. Here, we can transform x_1 and x_2 to $y_i = \alpha x_i + \beta, i = 1, 2$ and then considering the standard distance between these two transformed variables we find:

$$\begin{aligned} d(y_1, y_2) &= \frac{|y_1 - y_2|}{\sqrt{var(Y)}} \\ &= \frac{|\alpha(x_1 - x_2|)}{\sqrt{\alpha^2\sigma^2}} \\ &= d(x_1, x_2) \end{aligned}$$

The univariate standard distance has a straightforward generalisation to a multivariate setting. Considering now two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, with a common covariance matrix $\boldsymbol{\Sigma}$ the multivariate standard distance is given by:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_1, \boldsymbol{x}_2)}$$

Depending on whichever textbook is consulted, this multivariate standard distance may be referred to as the *statistical distance*, the *elliptical distance* or the *Mahalanobis distance*. Flury (1997) notes that the squared Mahalanobis distance $d(\boldsymbol{x}_1, \boldsymbol{x}_2)^2$ is sometimes simply referred to as the Mahalanobis distance, although it is not a valid distance measure. We refer here to the multivariate standard distance, $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ as the Mahalanobis distance, and where necessary, to $d(\boldsymbol{x}_1, \boldsymbol{x}_2)^2$ as the *squared* Mahalanobis distance.

It is worth noting that this measure was originally proposed by Mahalanobis (1930) as a measure of distance between two populations:

$$\Delta(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \sqrt{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)}$$

which has an obvious sample analogue as the distance between two mean vectors:

$$\Delta(\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2) = \sqrt{(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T \boldsymbol{S}^{-1} (\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2)}$$

where $\boldsymbol{S}$ is the pooled estimate of $\boldsymbol{\Sigma}$ given by $\boldsymbol{S} = \left[(n_1 - 1)\boldsymbol{S}_1 + (n_2 - 1)\boldsymbol{S}_2\right]/(n_1 + n_2 - 2)$.

Here, we are going to consider the distance between $\boldsymbol{x}$, a vector of random variables with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ and its mean:

$$\Delta(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}, \boldsymbol{\mu})}$$

and clearly we can find a sample analogue by estimating $\boldsymbol{\mu}$ by $\hat{\boldsymbol{x}}$ and $\boldsymbol{\Sigma}$ by $\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X}$. We note that in **R**, the `mahalanobis()` function is intended to returns the *squared* multivariate distance between a matrix $\boldsymbol{X}$ and a mean vector $\boldsymbol{\mu}$, given a user-supplied covariance matrix $\boldsymbol{\Sigma}$, i.e. we wish to calculate:

$$d(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}})^2 = (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})$$

We could also consider the Mahalanobis angle θ between two vectors at the origin:

$$\cos\theta = \frac{\boldsymbol{x}_1^T \boldsymbol{S}^{-1} \boldsymbol{x}_2}{d(\boldsymbol{x}_1, \boldsymbol{0}) d(\boldsymbol{x}_2, \boldsymbol{0})}$$

This can be extracted from within $\boldsymbol{R}$ using the following:

```
mahangle <- function(x1, x2, covmat){
  zero <- vector("numeric", length(x1) )
  num <- t(x1) %*% solve(covmat) %*% x2
  denom <- sqrt(mahalanobis(x1, zero, covmat)) *
     sqrt(mahalanobis(x2, zero, covmat))
  angle <- acos(num / denom)
  return(angle)
}
```

3.1.1 Distributional properties of the Mahalanobis distance

Remembering that where $z_1, \ldots, z_p \sim N(0, 1)$, if we form $y = \sum_{j=1}^{p} z_j^2$ then $y \sim \chi_p^2$ (Bilodeau and Brenner, 1999); for multivariate normal data, with p variables, the squared Mahalanobis distance can be considered against a χ_p^2 distribution:

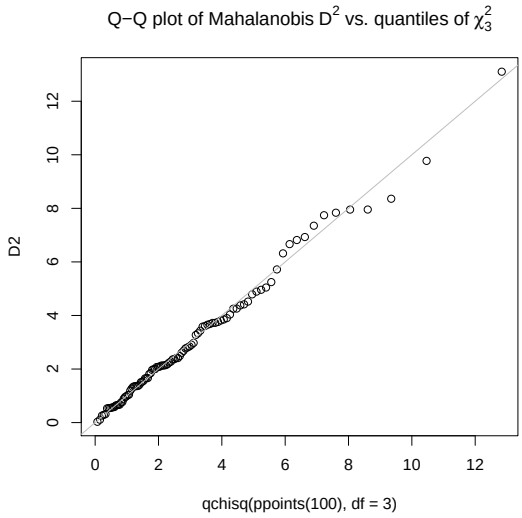


Figure 3.1: QQ plot of squared Mahalahobis distance plotted against χ^2 distribution

$$(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}) = \boldsymbol{z}^T \boldsymbol{z} \sim \chi^2_p \tag{3.1}$$

This immediately affords one method for assessing multivariate normality, quantiles of the Mahalanobis distance of $\boldsymbol{x}_i$, $i = 1, \ldots, n$ with respect to $\boldsymbol{\mu}$ can be plotted against quantiles of the χ^2_p distribution as an assessment of multivariate normality.

We can also define contours as a set of points of equal probility in terms of equal Mahalanobis distance:

$$(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T \hat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}) = \boldsymbol{z}^T \boldsymbol{z} = c^2 \tag{3.2}$$

for any constant $c > 0$. We will also find later in section 9.23.3 that the squared Mahalanobis distance is equivalent to the sum of squared prinicipal component scores. However, this chapter on distance is implicitly geared towards presentations in the chapter 4 on cluster analysis as well as chapter 5 on scaling methods. In that context it is worth noting that Mahalanobis distance is rarely used in cluster analysis, certainly Kendall (1975) points out its limitations in this context. Sporadic reports in the literature include Maronna and Jacovkis (1974) who report use of a particular clustering algorithm, k-means, with the Mahalanobis distance whereas Gnanadesikan et al. (1993) use it with hierarchical cluster analysis. This latter work may illustrate one of the difficulties in using the Mahalanobis distance in the requirement to assume a common covariance. However, whilst not proposing it's use in an automatic clustering algorithm, Atkinson et al. (2004) report use of Mahalanobis distance within the forward search to reliably identify subgroups within the data. They

propose a small modification to the Mahalanobis distance for use in cluster analysis as follows. The Mahalanobis distance is multiplied by $(|\hat{\boldsymbol{\Sigma}}_k^{-1}|^{1/2})^r$ for group k. Where $r = 0$ we have the usual distance, when $r = 1$ we have what they call the *standardised* Mahalanobis distance which eliminates the different variance between groups.

Having provided an overview of one distributionally important distance measure, before considering further measures we consider a few definitions. Flury (1997) notes that the squared Mahalanobis distance does not satisfy the axioms of distance.

3.2 Definitions

We now formalise our idea of a proximity measure. This term encapsulates both similarity and disssimilarity measures which have the obvious interpretation (measuring similarity and dissimilarity between entities), and can be found from each other by means of an appropriate monotonic transformation. We usually assume that these measures are symmetic.

A *distance* can be defined as a function $d(\cdot)$ that satisfies the following properties:

(1) Non-negative, that is $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ and

(2) Identified, that is $d(\boldsymbol{x}, \boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathbb{R}^p$;

(3) Symmetric, that is $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$;

In addition to satisfying these three properties, a *metric* also satisfies the following two properties:

(4) Definite, that is $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$;

(5) Triange inequality $d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) \geq d(\boldsymbol{x}, \boldsymbol{z})$ for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}$

It is worth noting that it is possible to compute a similarity measure, often denoted s, where $0 \leq S \leq 1$. A *similarity* function $s(\cdot, \cdot)$ satisfies (1) non-negativity $s(\boldsymbol{x}, \boldsymbol{y}) \geq 0$, (2) symmetry $S(\boldsymbol{x}, \boldsymbol{y}) = s(\boldsymbol{y}, \boldsymbol{x})$ as well as:

(3) $s(\boldsymbol{x}, \boldsymbol{y})$ increases in a monotone fashion as $\boldsymbol{x}$ and $\boldsymbol{y}$ become more similar. A *dissimilarity* function satisfies the first two but clearly 3 is reversed, i.e. it decreases as $\boldsymbol{x}$ and $\boldsymbol{y}$ become more similar.

Dissimilarity is the opposite of similarity, therefore any monotonically decreasing transformation of s can provide a dissimilarity measure. The most obvious tranform would be to take $d = 1 - s$ but we will consider a few alternatives later.

3.3 Distance between points

Two **R** packages are needed to provide most of the distance functions considered here. In addition to the default `stats` library, which provides the `dist()` function, we require the `cluster` package for the `daisy()` function. Some further correlation based measures can be found in the `Dist()` function in the `amap` package as well as `BioBase` from Bioconductor.

We now consider a range of ways in which a multivariate distance can be measured. In conducting an analysis, some decision needs to be made as to whether to scale variables, or whether to remove highly correlated variables from the analysis. For example, Gnanadesikan (1997) gives an artificial example which illustrates how rescaling variables can subsequently alter impression of groupings.

3.3.1 Quantitative variables - Interval scaled

It is reasonably straightforward to suggest a number of dissmilarity measures d_{ij} which measure the distance between individual i and j.

Euclidean distance

.

The Euclidean distance, or the l_2 norm, is perhaps the most commonly used distance measure. As mentioned in section 3.1, this distance could be considered simply as the Mahalanobis distance where $\sigma = 1$. Especially in the context of cluster analysis, where we hope to identify distinct sub-groups within the data, it is not clear how we might determine the covariance matrix hence the Mahalanobis distance has seen little use. The Euclidean distance, which is quite simply the square root of the squared distance between any two vectors, which can be quite simply interpreted as the physical distance between two p-dimensional points is also a convenient measure to understand. Formally, we can express this measure as:

$$d_{ij} = \left(\Sigma_{k=1}^{p}(x_{ik} - x_{jk})^2\right)^{\frac{1}{2}}$$

where we are trying to measure the distance between observations in row i and row j, in other words x_{ik} is the kth observation in row i, and x_{jk} is the corresponding kth observation in row j. Euclidean distance can be readily calculated in **R** using the `dist()` function with the default `method = "euclidean"`, as well as by `daisy()` with the default `metric = "euclidean"`, although in `daisy()` it is possible to standardise the data within the calculations by adding `stand = TRUE` to the function call.

Scaled Euclidean distance

It is possible to introduce a suitable weight w_k such as the inverse of the standard deviation of the kth variable, i.e. $w_k = s_k^{-1}$, or even the inverse of the range of the data.

$$d_{ij} = \sqrt{(\Sigma_{k=1}^{p} w_k^2 (x_{ik} - x_{jk})^2)}$$

No explicit routines are available to compute this measure, but clearly if the co-ordinates are rescaled by $\sqrt{w_k}$ this can be calculated implicitly.

City Block metric

The City Block metric, formally referred to as an l_1 norm, measures the absolute difference between two vectors. It is so-named because it measures the distance between two points in terms of movements parallel to the axis and therefore resembles the distance between two points in a city. Krause (1975) (who had obviously never been in a London taxi) called this distance the *taxicab* distance, Brandeau and Chiu (1988) used the term *rectilinear*, but perhaps the most common alternative name is *Manhattan*, suggested by Larson and Sadiq (1983) reflecting the famous city block layout in Manhattan. Formally, we can express this distance as:

$$d_{ij} = (\Sigma_{k=1}^{p} |x_{ik} - x_{jk}|)$$

It can be calculated in R using the `dist()` function with `method = "manhattan"`

Minkowski metric

The Minkowski metric, or the l_r norm, is a generalisation of the Manhattan and Euclidean distances.

$$d_{ij} = \left(\Sigma_{k=1}^{p} |x_{ik} - x_{jk}|^{\lambda}\right)^{1/\lambda}$$

Where $\lambda = 1$ we have the Manhattan metric, where $\lambda = 2$ we have the Euclidean distance. It can be noted that increasing λ exaggerates dissimilar units relative to similar ones. This metric can be calculated in R using the `dist()` function with `method = "minkowski"` but additionally requires an argument to `p` to set λ, the power of this distance. Therefore, for example `dist(x, method = "minkowski", p=2)` gives the Euclidean distance for matrix `x`.

Canberra metric

The Canberra metric (Lance and Williams, 1966) can be regarded as a generalisation of binary dissimilarity measures, and is very sensitive to small changes close to $x_{ik} = x_{jk} = 0$. It can be scaled by division by p, the number of variables to ensure it lies in the range (0,1). Terms with zero numerator and denominator are omitted from the sum and treated as if the values were missing.

$$d_{ij} = \begin{cases} 0 & for\ x_{ik} = x_{jk} = 0 \\ \Sigma\left(\frac{|x_{ik}-x_{jk}|}{|x_{ik}+x_{jk}|}\right) & for\ x_{ik} \neq 0\ or\ x_{jk} \neq 0 \end{cases}$$

This metric can be calculated in **R** using the `dist()` function with `method = "canberra"`

Czekanowski Coefficient

Finally, we mention the Czekanowski Coefficient, which for continuous variables can be given as:

$$d_{ij} = 1 - \frac{2\sum_{k=1}^{p} min(x_{ik}, x_{jk})}{\sum_{k=1}^{p}(x_{ik} + x_{jk})}$$

3.3.2 Distance between variables

We next consider a number of correlation based distance measures. Note that when used conventionally for calculating the correlation between two variables we work with standardised columns. In order to measure the similarity between two individuals we must therefore work with standardised rows, this may not be a sensible procedure. For example, if variables are measured on different scales the idea of a row mean may not be clear. There is further material in the literature questioning the use of these measures (Jardine and Sibson, 1971; Fleiss and Zubin, 1969) and Everitt et al. (2001) note that correlation measures cannot distinguish the size of two different observations, giving the example $\boldsymbol{x}_1^T = c(1, 2, 3)$ and $\boldsymbol{x}_2^T = c(1, 2, 3)$ have correlation $\rho_{12} = 1$ yet $\boldsymbol{x}_2^T$ is three times the size of $\boldsymbol{x}_1^T$. Nevertheless, correlation based measures have become particular popular in a bio-informatics setting where some of the noted limitations do not apply (all variables are measured on a comparable scale) and in fact it is not always clear what a row and a column mean in that application area.

We therefore consider four four distances that can be obtained a correlation measure. Some thought needs to be given to determining the transformation from a correlation coefficient to a distance measure. The Pearson correlation coefficient is defined in the range $-1 \leq \rho_{ij} \leq 1$. Everitt et al. (2001) suggest using $d_{ij} = \frac{1-\rho_{ij}}{2}$. Gentleman et al. (2005) suggest that it may be appropriate under some circumstances to use the absolute value of the correlation, that is $d_{ij} = 1 - |\rho_{ij}|$ which

means that there will be little distance between rows having strong positive and strong negative correlation. In terms of measuring the dissimilarity between variables, Krzanowski (2000) suggests a further alternative using $d_{ij} = 1 - (\rho_{ij})^2$. Examining pre-Bioinformatics data, Lance and Williams (1979) who compared a number of transformations and expressed a strong preference for the first transformation, and a strong disdain for the third.

It should be noted that these measures are quite badly affected by outliers. As a result, non-parametric versions may often be preferred. Conversely, these measures are invariant to change of location or scale transformation which is rather useful. It should be noted in bio-informatics practice that they tend to group genes whose expression patterns are linearly related, there is some empirical support from that application for their use in a particular context.

Pearson correlation distance

$$d(x_{ij}, x_{ik}) = 1 - \rho_{ij} = 1 - \frac{\sum_{i=1}^{p}(x_{ij} - \bar{x}_{\cdot j})(x_{ik} - \bar{x}_{\cdot k}}{\sqrt{\sum_{i=1}^{p}(x_{ij} - \bar{x}_{\cdot j})^2 \sum_{i=1}^{p}(x_{ik} - \bar{x}_{\cdot k})^2}}$$

Where data are scaled, i.e. mean centred and standardised by the variance so that $\boldsymbol{x}_{\cdot j}$ and $\boldsymbol{x}_{\cdot k}$ are p variable vecotres with zero mean and unit variance the relationship between the Euclidean distance and the Pearson correlation is given by:

$$d_{ij}^{(Euclidean)} = \sqrt{2p(1 - \rho_{ij})}$$

The pearson based distance measure can apparently be obtained from `Dist()` in the `amap` package, where it is referred to as the "Centred Pearson" by specifying `method = "correlation"` in the function call.

Cosine correlation coefficient

This is similar to the Pearson Correlation coefficient based distance measure but without the mean standardisation

$$d(x_{ij}, x_{ik}) = 1 - \frac{\boldsymbol{x}_{\cdot j}^T \boldsymbol{x}_{\cdot k}}{||\boldsymbol{x}_{\cdot j}|| ||\boldsymbol{x}_{\cdot k}||} = 1 - \frac{|\sum_{i=1}^{p}(x_{ij})x_{ik}}{\sqrt{\sum_{i=1}^{p} x_{ij}^2 \sum_{i=1}^{p} x_{ik}^2}}$$

The cosine correlation based distance measure, referred to as the "Not-centred Pearson can be obtained from `Dist()` in the `amap` package by specifying `method = "pearson"` in the function call. It is not clear from the help file how the correlation measure is transformed into a distance measure.

Spearman sample correlation distance

$$d(x_{ij}, x_{ik}) = 1 - \rho_{ij} = 1 - \frac{\sum_{i=1}^{p}(rank(x)_{ij} - rank(\bar{x})_{.j})(rank(x)_{ik} - rank(\bar{x})_{.k})}{\sqrt{\sum_{i=1}^{p}(rank(x)_{ij} - rank(\bar{x})_{.j})^2 \sum_{i=1}^{p}(rank(x)_{ik} - rank(\bar{x})_{.k})^2}}$$

This requires `spearman.dist()` in package `bioDist`, and can be computed via `Dist()` in the `amap` package with a call containing `method="spearman"`.

Kendall's τ sample correlation distance

$$d(x_{ij}, x_{ik}) = 1 - \tau_{ij} = 1 - \frac{\sum_{i=1}^{p} sign(x_{ij} - \bar{x}_{.j}) sign(x_{ik} - \bar{x}_{.k})}{p(p-1)}$$

This requires `tau.dist()` in package `bioDist`

3.3.3 Quantitative variables: Ratio Scaled

Kaufman and Rousseeuw (1989) briefly discuss ratio scaled variables, and give examples including micro-organism growth which follows and exponential power law. Clearly, we could just consider these as interval scale variables and use any of the previous measures. They discuss the possibility of taking a logarithmic transformation of such data where it may be appropriate, obviously having the exponential growth application in mind. The logarithmic transformation can be dealt with in `daisy()` by using the `type="logratio"` command. Alternatively, it would be possible to treat such variables as being continuous ordinal data and use rank-based non-parametric procedures. As discussed further in section 3.3.5, using the `type="ordratio"` command within `daisy()` generates standardised variates from the ranks which are subsequently analysed with a scaled City Block metric, alternativly, the two non-parameteric correlation derived measures described in section 3.3.2 and 3.3.2 may also be useful.

3.3.4 Dichotomous data

Where x_{ik} can only take one of two values, these are coded as 0 and 1:

		Object Two	
		1	0
Object 2	1	a	b
	0	c	d

where $p = a + b + c + d$, some common dissimilarity measures are:

In this table, a denotes an agreement (both objects have a zero in the same position), d shows an agreement where both objects have a one, c and d denote the two possible disagreements. We should firstly comment in more detail on the nature of dichotomous data. Gower (1971) distinguishes two types of binary variables, symmetric and assymetric. Binary variables such as gender (male and female) or handedness (left or right) are clearly symmetric and the distance measure should not change depending on the way we code these two levels as 0 and 1. In other words, a and d should act the same way in the table.

We can therefore consider the following symmetric measures.

(Based on the) simple matching coefficient

The *simple matching coefficient*, also known as the *M-coefficient* or the *affinity index*, is quite simply the proportion of variables in agreement in two objects. The distance measure is found by subtracting this value from 1 (or calculating the proportion of disagreements):

$$d_{ij} = 1 - \frac{a+d}{a+b+c+d} = \frac{b+c}{a+b+c+d} \tag{3.3}$$

This measure can be calculated in `daisy()` by providing a list indicating those variables to be regarded as symmetric, i.e. `(list("symm", "symm", "symm")`. It may be noted in passing that if we force a calculation of Manhattan distance we get estimate $b+c$ we omit standardisation and simply calculate the sum of disagreements. Also, the Euclidean distance is the square root of the dissimilarity derived from the simple matching coefficient. Two further symmetric measures include **?** which doubles the weight of the disagreements:

$$d_{ij} = 1 - \frac{a+d}{(a+d)+2(b+c)} = \frac{2(b+c)}{(a+d)+2(b+c)}$$

and the Sokal and Sneath (1963) measure which doubles the weight of the agreements:

$$d_{ij} = 1 - \frac{2(a+d)}{2(a+d)+(b+c)} = \frac{b+c}{2(a+d)+(b+c)}$$

All three measures are monotonically related and there seems little imperative to use anything other than the simple matching coefficient based dissimilarity measure. Life does however get rather more interesting if we want to work with assymetric binary variables. Some care is needed in analysis in determining whether binary variables are symmetric or assymetric. A classical example would concern variables measuring presence or absence. The thought is that if two individuals share the presence of some attribute we can consider them similar, but if they share the absence of an attribute we

do not know whether they can be considered similar. For example, if we collect data on individuals who travelled to a particular location, we can consider them similar if they both drove by car, but if neither drove by car it is clear there are a range of reasons, which could include not owning a car, preferring another form of transport, living within walking distance and so on.

Jaccard coefficient

Perhaps the most common assymetric measure of distance is the Jaccard Coefficient Sneath (1957), which measures the proportion of agreements on the variable coded 1 among all such agreements and disagreements (i.e. ignoring all possible agreements on variable coded 0). Formally, this can be set out as:

$$d_{ij} = 1 - \frac{a}{a+b+c} = \frac{b+c}{a+b+c}$$

This seems to be the value calcuated by **R**, when `method="binary"` is used in the call to `dist()`, it is also available in `daisy()` when the a list is supplied which indicates those variables to be considered as binary assymetric variables, i.e. `list("asym", "asym")`

As with symmetric measures, there are a few alternatives which alter the weightings.

Czekanowski coefficient

The Czekanowski coefficient (Dice, 1945) increases the weight of the agreements

$$d_{ij} = 1 - \frac{2a}{2a+b+c} = \frac{b+c}{2a+b+c}$$

whearas the Sokal and Sneath (1963) coefficient increases the weight of the disagreements:

$$d_{ij} = 1 - \frac{a}{a+2(b+c)} = \frac{2(b+c)}{a+2(b+c)}$$

We extract a small part of an example given by Kaufman and Rousseeuw (1989) to illustrate the non-monotonicity of the symmetric and asymmetric measures.

		Variable 1	
		+	-
Variable	+	a	b
	-	c	d

Name	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Ilan	1	0	1	1	0	0	1	0	0	0
Jacqueline	0	1	0	0	1	0	0	0	0	0
Lieve	0	1	0	0	0	0	0	1	1	0
Peter	1	1	0	0	1	0	1	1	0	0

where x_1 = Sex(Male = 1, Female = 0), x_2 = Married(Yes = 1, No = 0), x_3 = Hair(Fair = 1, Dark = 1), x_4 = Eyes (Blue = 1, Brown = 0), x_5 = Wears Glasses(Yes = 1, No = 1), x_6 = Face (Round = 1, Oval = 0), x_7 = Outlook(Pessimist = 1, Optimist = 0), x_8 = Type(Evening = 1, Morning = 0) x_9 = Only Child (1 = Yes, 0 = No) x_{10} = Handedness (1 = Left, 0 = Right).

Using the symmetric, simple matching coefficient based distance measure they note that:

$$d(Jacqueline, Lieve) = 0.300 \; d(Ila, Peter) = 0.500$$

whereas for the asymmetric, Jaccard coefficient we have:

$$d(Jacqueline, Lieve) = 0.750 \; d(Ila, Peter) = 0.714$$

Although Kaufman and Rousseeuw (1989) state that the Jaccard coefficient is inappropriate, it could be argued that some of these variables are assymetric (there are a variety of reasons why someone might record that they were not-married). Nevertheless, the point of their illustration was to highight the non-monotonicity. Whilst we expect the measures to be different, note that for the symmetric coefficient $d(Jacqueline, Lieve) < d(Ila, Peter)$, whereas for the assymetric coefficient $d(Jacqueline, Lieve) > d(Ila, Peter)$.

Similarities between variables

$$\chi^2 = \frac{(ad - bc)^2(a + b + c + d)}{(a + b)(a + c)(c + d)(b + d)}$$

which may require some standardisation:

$$d_{kl} = 1 - \sqrt{\frac{\chi^2}{a+b+c+d}}$$

3.3.5 Qualitative variables

Following Kaufman and Rousseeuw (1989) we consider a variable where we have $m = 1, \ldots, M$ states. It would be possible to create a set of M binary variables, with 0 indicating absence of a particular category within a variable and 1 indicating presence. Alternatively, a nominal variable could be collapsed in some suitable manner. However, Sokal and Michener (1958) suggest a simple matching coefficient, a corresponding distance can be found by substracting this from 1. Denoting the number of variables on which objects i and i agree by u, and the total number of variables by p this can be expressed as:

$$d(x_{ij}, x_{ik}) = 1 - \frac{u}{p} = \frac{p-u}{p}$$

This measure is invariant to the codings used or the order of the variables, and can be extended in the same way as that suggested for binary variables by Rogers and Tanimoto (1960) and Sokal and Sneath (1963) by doubling the weight of disagreements and agreements respectively. Kaufman and Rousseeuw (1989) review proposals to weight the measure depending on the size of M.

The simple matching coefficient is available in **R** by using `daisy()` having specified that the variable concerned is a factor, by ensuring the elements of `x` supplied to the function have class `factor`.

It's also obvious that such variables can be ordered, and also that ordered variables may be derived from continuous data. We can either obtain the ranks and treat the ranks as continuous variables applying any of the quantitative distance measures discussed above. A possible derivation, having first scaled the ranks is given by:

$$z_{ij} = \frac{r_{ij} - 1}{M_j - 1}$$

When using `daisy()`, if a discrete variable has the class set to `"ordered"`, or if a continuous variable is supplied with the argument `type = "ordratio"` z_{ij} will be computed as in figure 3.3.5 and treated as a continuous variable. Distance will subsequently be computed by means of the City Block distance, which will be scaled by the number of such variables analysed.

Alternative, one of the non-parametric correlation measures in section 3.3.2 or section 3.3.2 could be used, especially where we are measuring distance between variables rather than between individuals.

3.3.6 Different variable types

Finally, we consider the possibility that a particular data set contains a variety of variable types. It might be possible to treat all variables as interval scaled continuous variables, or somehow recode them all as binary or ordinal variables. It may be better to find some way of combining distance that has been measured in the most appropriate way for each type of variable. As a result, possibly the most popular method for measuring dissimilarity in this situation has been derived from Gower's coefficient of similarity Gower (1971). In its original incarnation, this measure could combine interval, nominal and binary data. Consider the following, where we basically sum the individual similarities however calculated and divide them by the total number of applicable comparisons:

$$d(x_{ij}, x_{jk}) = 1 - \frac{\sum_{k=1}^{p} \delta_{ijk} s_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}} \tag{3.4}$$

The indicator δ_{ijk} is set to 1 when both measurements for x_{ij} and x_{ik} are non-missing, it is zero otherwise. It is also zero for binary variables where there is a $0-0$ match, i.e. the original measure assumed assymetric dichotomous variables. We briefly consider how similarities for each of the three variable types is calculated:

- Quantitative (interval scaled) variables.

  The similarity measure is given by:

  $$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{range\ of\ variable\ k} \tag{3.5}$$

  This is essentially the City Block distance with the extra assumption that all variables had first been standardised by dividing by their range. If there are mixed variables within the data frame or matrix `x` supplied to `daisy()`, this standardisation is applied by default (regardless of any arguments supplied to `stand`.

- Qualitative (nominal) variables. These are derived from the simple matching coefficient, the similarity is therefore the proportion of matches among all possible matches:

  $$s_{ijk} = \begin{cases} 1 \text{ if i and i agree on variable k} \\ 0 \text{ otherwise} \end{cases}$$

- Dichotomous variables The original incantation assumed asymmetric variables, hence the Jaccard coeffcient is used. If we consider $k = 1, \ldots, 4$ variables for individuals i and j, we can see the possible outcomes:

i	1	1	0	0
j	1	0	1	0
s_{ijk}	1	0	0	0
δ_{ijk}	1	1	1	0

The original measure has been extended by Kaufman and Rousseeuw (1989) (who set out the calculations as distances rather than similarities) to incorporate symmetric binary and ordinal and ratio variables. Ordinal variables are ranked, and the ranks used in 3.5, ratio variables are either ranked or logged and then 3.5 is used. As has been noted earlier, this is achieved by setting the class of the variables to be "numeric", "factor" or "ordered", or providing the arguments `type = "asymm", "symm", "ordratio", "logratio"` to estimate appropriate measures for asymetric binary variables (Jaccard), symmetric binary, ordinal ratio variables or log transformed ratio variables respectively.

It should be noted that Gower (1971) shows, provided there are no missing values the $n \times n$ similarity matrix obtained from an $n \times p$ data matrix $\boldsymbol{X}$ is positive semi-definite. If we obtain a dissimilarity matrix from this measure using $d(x_{ik}, x_{jk}) = \sqrt{(1 - s(x_{ik}, x_{jk})}$ the resultant matrix:

$$\Delta = \begin{pmatrix} 0 & d(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & d(\boldsymbol{x}_1, \boldsymbol{x}_p) \\ d(\boldsymbol{x}_2, \boldsymbol{x}_1) & 0 & \cdots & d(\boldsymbol{x}_2, \boldsymbol{x}_p) \\ \vdots & \vdots & \cdots & \vdots \\ d(\boldsymbol{x}_p, \boldsymbol{x}_1) & d(\boldsymbol{x}_p, \boldsymbol{x}_2) & \cdots & 0 \end{pmatrix}$$

is Euclidean. We will next consider this important property of proximity matrices, it will particularly inform later developments in terms of metric scaling.

3.4 Properties of proximity matrices

A few words are placed here concerning proximity matrices, the $n \times n$ matrix comparing every individual with each other. It is possible that the only information available in a particular study is such a matrix, as happens with sensory experiments or the rather famous study comparing matched judgements on morse code characters (Rothkopf, 1957). Some properties of these matrices will be important in later developments, particularly scaling as discussed in chapter 5. Earlier, in section 3.3.6, we rather glibly stated that a dissimilarity matrix obtained from Gower's coefficient of similarity can be Euclidean. As might be anticipated, a matrix where the elements have been derived from the Euclidean distance (section 3.3.1) is also Euclidean, but the concept requires further examination. Gower (1966) demonstrated that Euclidean properties could be met by transforming for the elements $s(x_i, x_j)$ of a similarity matrix $\boldsymbol{S}$ to the elements $d(x_i, x_j)$ of a dissimilarity matrix $\boldsymbol{D}$:

$$d(x_i, x_j) = \sqrt{1 - s(x_i, x_j)}$$

Denoting the distance between two points by $d(x_i, x_j)$ and the proximity between two individuals by $\delta(x_i, x_j)$, we are ultimately interested in the proximity matrix $\boldsymbol{\Delta}$

When forming a matrix $\boldsymbol{\Delta}$ from a dissimilarity matrix $\boldsymbol{D}$, Lingoes (1971) suggested tranforming the elements by $\delta(x_i, x_j) = \sqrt{d(x_i, x_j)^2 + c_1}$, Cailliez (1983) suggested $\delta(x_i, x_j) = d(x_i, x_j) + c_1$, both providing methods for finding the constants c_1 and c_2.

In general, when considering the $n \times n$ dissimilarity matrix, Δ, containing elements $d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})$. This matrix can be considered Euclidean if the n individuals can be represented as points in space such that the Euclidean distance between points i and j is $d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})$. In general, $\boldsymbol{\Delta}$ is Euclidean if and only if the following matrix is positive semi-definite:

$$(\boldsymbol{I} - \mathbf{1}\boldsymbol{s}^T)\boldsymbol{\Gamma}(\boldsymbol{I} - \mathbf{1}\boldsymbol{s}^T)$$

where $\boldsymbol{\Gamma}$ has elements $\frac{1}{2}d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})^2$, $\boldsymbol{I}$ is the identity matrix, $\mathbf{1}$ is a vector of n ones and $\boldsymbol{s}$ is an n-element vector such that $\boldsymbol{s}^T\mathbf{1} = 1$.

Important special cases are where $\boldsymbol{s} = \frac{1}{n}\mathbf{1}$ which centres at the origin (proof is given in Mardia et al. (1979) regarding the Euclidian property) and where $\boldsymbol{s}$ which has 1 in its ith position and 0 elsewhere (proof is given by Gower (1984) of the Euclidean property). The significance of the Euclidean property will be discussed in chapter 5 when we consider scaling methods.

In addition to the Euclidean property, a matrix can be considered *metric*, if the metric inequality holds for all triplets i, j, k within $\boldsymbol{\Delta}$

$$d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot}) + d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{k\cdot}) \geq d(\boldsymbol{x}_{j\cdot}, \boldsymbol{x}_{k\cdot})d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot}) \tag{3.6}$$

If $\boldsymbol{Delta}$ is metric, then so are matrices with elements $d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})+c^2$ as well as $d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})/(d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})+ c^2)$ where c is any real constant and $i \neq j$, as is $d(\boldsymbol{x}_{i\cdot}, \boldsymbol{x}_{j\cdot})^{1/r}$ for $r \geq 1$ with again $i \neq j$. In the case that $\boldsymbol{\Delta}$ is non-metric, Krzanowski and Marriott (1994a) indicate that where $c \geq max_{i,j,k}|\delta(x_i, x_j) + \delta(x_i, x_k) - \delta(x_j, x_k)|$, the matrix with elements $\delta(x_i, x_j) + c$ is metric.

Again, the significance of the metric property will be discussed in chapter 5 when we consider scaling methods.

Chapter 4

Cluster analysis

Cluster "analysis" describes a range of algorithms for investigating structure in data, the main interest is in finding groups of objects who are more alike. A large number of books are dedicated to this one subject, for example Kaufman and Rousseeuw (1989); Everitt et al. (2001); Gordon (1999), the former book supporting some rather interesting code that has been ported to **R** (the S-PLUS version is described in Struyf et al. (1997)). It may be worth noting that some multivariate authors do not cover it at all Flury (1997) preferring a formulation based upon mixture models, and (Venables and B.D.Ripley, 2002, page 316) indicate a preference for using visualisation methods for finding such structure in data. Whilst cluster analysis is most often directed towards finding subsets of individuals that are more alike than other subsets, it is worth noting that variable clustering is also possible. The heatmap in figure 4.1 has a dendrogram for both individuals and variables.

```
x  <- as.matrix(mtcars)
rc <- rainbow(nrow(x), start=0, end=.3)
cc <- rainbow(ncol(x), start=0, end=.3)
hv <- heatmap(x, col = cm.colors(256), scale="column",
     RowSideColors = rc, ColSideColors = cc, margin=c(5,10),
     xlab = "specification variables", ylab= "Car Models",
     main = "Heatmap of mtcars data")
```

Modern computing facilities have widened the possibilities for visualisation considerable (both in terms of linked displays as well as the ability to numerically optimise various projection criteria). However, when investigating the results of any scaling or projection methods there may still be interesting structures within the data. The interest in cluster analysis lies in finding groups within the data. When the objects within a group are very similar this can be described as internal cohesion (or homogeneity), when they is a large dissimilarity between groups this can be referred to as external separation (or isolation). Where we have internal cohesion and external separation, one usually refers to the situation as "clustering". If the objects have been more-or-less aribtrarily divided into groups

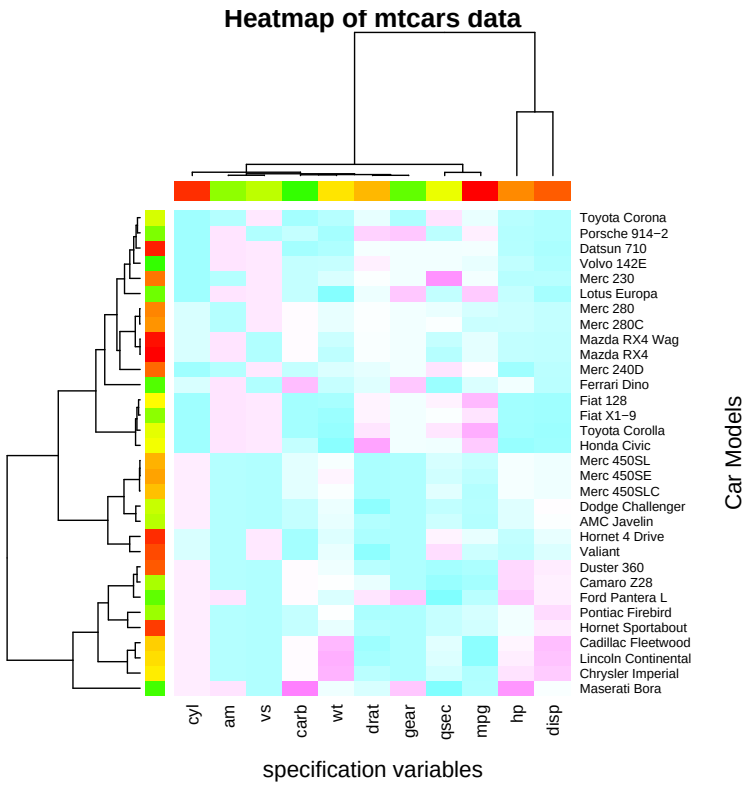


Figure 4.1: Heatmap of `mtcars` data; variables scaled in columns

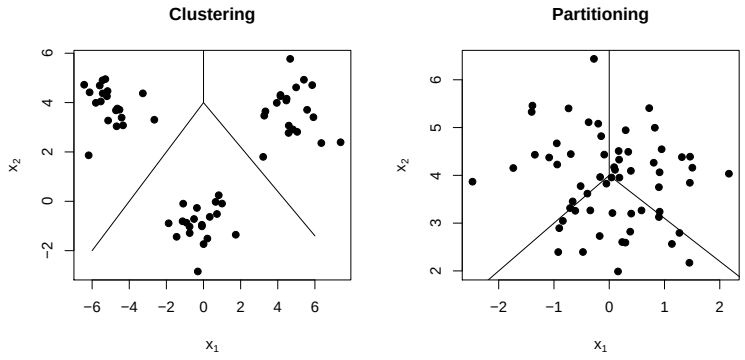


Figure 4.2: Artificial data suggesting a difference between "clustering" and "dissecting"

we could refer to this as "dissection". Clustering and dissection may both the useful in different contexts (e.g. taxonomy and market research). One attempt to illustrate this concept is given in figure 4.2.

There are a wide range of algorithms that have been developed to investigate clustering within data. These can be considered in a number of ways:

- Hierarchical Methods
  - Agglomerative clustering (`hclust()`, `agnes()`)
  - Divisive clustering (`diana()`, `mona()`)
- Partitioning methods (`kmeans()`, `pam()`, `clara()`)

Hierarchical clustering provides a set of clusterings, either for $k = n, \ldots, 2$ (agglomerative) or $k = 2, \ldots, l$ (divisive). The clustering is represented in a dendrogram and can be cut at arbitrary heights to give a fixed number of clusters. It perhaps has a logical interpretation in numerical taxonomy. Partitioning methods usually work by assuming a pre-determined number of clusters k (although it is obviously possible to explore solutions over a range of values of k). As (Seber, 1984, page 379) points out, the number ofways $S_{k,n}$ of partitioning n objects into k groups, given by:

$$S_{k,n} = \frac{1}{k!}\sum_{j=1}^{k}\left(\begin{array}{c} k \\ j \end{array}\right)(-1)^{k-j}j^n \approx_{n\to\infty} \frac{k^n}{k!}$$

is a second type Stirling number, and where k is not specified we have $\sum_{k=1}^{K} S_{k,n}$ partitions.

For $n = 50$ and $k = 2$ this is in the order of 6×10^{29}

4.1 Introduction to agglomerative hierarchical cluster analysis

Hierarchical cluster analysis finds representations that obey the ultrametric inequality.

$$d(x_{ij}, x_{ik}) \leq max(d(x_{ij}, x_{il})d(x_{il}, x_{ik}))$$

We will work through three conventional examples of cluster analysis manually before considering an **R** implementation in more detail. For the sake of the demonstations, consider the following distance matrix:

	a	b	c	d	e
a	0				
b	2	0			
c	6	5	0		
d	10	9	4	0	
e	9	8	5	3	0

Initially, each individual is put in its own cluster. Subsequently, at each stage individuals are joined to other individuals to form clusters, and the distance measure between a units can be readjusted in some way. Then similar clusters are joined until all individuals have been joined.

4.1.1 Nearest neighbour / Single Linkage

This can be obtained in **R** using the `method = "single"` instruction in the call to `hclust()`, where it suggests that this method finds "friends of friends" to join each cluster in a similar way to that used in minimum spanning trees.

The decision to merge groups is based on the distance of the nearest member of the group to the nearest other object. Clearly, with a distance of 2, inviduals a and b are the most similar.

	a	b	c	d	e
a	0				
b	2	0			
c	6	5	0		
d	10	9	4	0	
e	9	8	5	3	0

We therefore merge these into a cluster at level 2:

Distance	Groups
0	a b c d e
2	(ab) c d e

and we now need to re-write our distance matrix, whereby:

$$d_{(ab)c)} = min(d_{ac}, d_{bc}) = d_{bc} = 5$$
$$d_{(ab)d)} = min(d_{ad}, d_{bd}) = d_{bd} = 9$$
$$d_{(ab)e)} = min(d_{ae}, d_{be}) = d_{be} = 8$$

This gives us a new distance matrix

	(ab)	c	d	e
(ab)	0			
c	5	0		
d	9	4	0	
e	8	5	3	0

Now we find that the two nearest objects are d and e, these can be merged into a cluster at height 3:

Distance	Groups
0	$a\ b\ c\ d\ e$
2	$(ab)\ c\ d\ e$
3	$(ab)\ c\ (de)$

We now need to find the minimum distance from d and e to the other objects and reform the distance matrix:

	(ab)	c	(de)
(ab)	0		
c	5	0	
(de)	8	4	0

Clearly, the next merger is between (de) and c, at a height of 4.

Distance	Groups
0	$a\ b\ c\ d\ e$
2	$(ab)\ c\ d\ e$
3	$(ab)\ c\ (de)$
4	$(ab)\ (cde)$

And it is also clear that the next step will involve merging at a height of 5.

Distance	Groups
0	$a\ b\ c\ d\ e$
2	$(ab)\ c\ d\ e$
3	$(ab)\ c\ (de)$
4	$(ab)\ (cde)$
5	$(abcde)$

The corresponding dendrogram is illustrated in figure 4.3.

4.1.2 Furthest neighbour / Complete linkage

Obtained in **R** using the `method = "complete"` instruction in the call to `hclust()`, where it suggests that this method finds similar clusters. Complete linkage methods tend to find similar clusters.

Groups are merged when the furthest member of the group is close enough to the new object.

	a	b	c	d	e
a	0				
b	2	0			
c	6	5	0		
d	10	9	4	0	
e	9	8	5	3	0

Start assembling details on the distances; we start as before

Distance	Groups
0	a b c d e
2	(ab) c d e

However, the reduced distance matrix will be different:

$$d_{(ab)c)} = max(d_{ac}, d_{bc}) = d_{ac} = 6$$
$$d_{(ab)d)} = max(d_{ad}, d_{bd}) = d_{ad} = 10$$
$$d_{(ab)e)} = max(d_{ae}, d_{be}) = d_{ae} = 9$$

	(ab)	c	d	e
(ab)	0			
c	6	0		
d	10	4	0	
e	9	5	3	0

Although the next step will be identical (we merge d and e at a height of 3)

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)

We now need to find the minimum distance from d and e to the other objects and reform the distance matrix:

	(ab)	c	(de)
(ab)	0		
c	6	0	
(de)	10	5	0

Although we are still going to merge (de) and c note that the height is different being 5.

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)
5	(ab) (cde)

So our final merge will take place at height 10.

	(ab)	(cde)
(ab)	0	
(cde)	10	0

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)
5	(ab) (cde)
10	(abcde)

In this simple demonstration, the dendrogram, illustrated in the centre of figure 4.3 obtained is similar in shape, but all the merges are at different height.

4.1.3 Group average link

This requires `agnes()` in package `cluster`, called with the `method="average"` instruction.

This time we merge two groups is the average distance between them is small enough.

	a	b	c	d	e
a	0				
b	2	0			
c	6	5	0		
d	10	9	4	0	
e	9	8	5	3	0

Start assembling details on the distances; again, we start as before:

Distance	Groups
0	a b c d e
2	(ab) c d e

But the reduced distance matrix will be different again:

$$d_{(ab)c)} = (d_{ac} + d_{bc})/2 = 5.5$$
$$d_{(ab)d)} = (d_{ad} + d_{bd})/2 = 9.5$$
$$d_{(ab)e)} = (d_{ae} + d_{be})/2 = 8.5$$

	(ab)	c	d	e
(ab)	0			
c	5.5	0		
d	9.5	4	0	
e	8.5	5	3	0

Yet again, the next merge step will be identical (we merge d and e, only they are merged at height 3)

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)

We now need to find the minimum distance from d and e to the other objects and reform the distance matrix:

	(ab)	c	(de)
(ab)	0		
c	5.5	0	
(de)	9	4.5	0

Again, we still going to merge (de) and c note that the height is different (4.5)

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)
5	(ab) (cde)

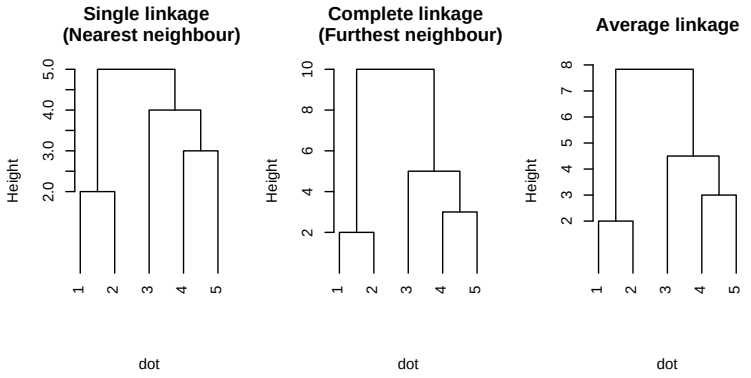


Figure 4.3: Dendrograms from three basic cluster methods

	(ab)	(cde)
(ab)	0	
(cde)	7.8	0

So our final merge will take place at height 7.8.

Distance	Groups
0	a b c d e
2	(ab) c d e
3	(ab) c (de)
4.5	(ab) (cde)
7.8	(abcde)

In this simple demonstration, the dendrogram obtained is similar in shape, but all the merges are at different height. This dendrogram is depicted on the right of figure 4.3. Whilst the role of the dendrogram seems obvious, it should be acknowledged that there are some algorithmic details needed to determine how to present these. We will consider other clustering methods shortly, but it should be noted that the centroid and median approaches can lead to reversals in the dendrogram.

4.1.4 Alternative methods for hierarchical cluster analysis

Given that cluster "analysis" is essentially an algorithmically guided exploratory data analysis it is perhaps no surprise, firstly that there have been many other methods proposed and secondly that there have been attempts to generalise the algorithm. Lance and Williams (1966, 1967) proposed a general recurrence formula which gives the distance between a newly amalgated group $C_k \bigcup C_l$ and some other group C_m:

$$d_{C_k \bigcup C_l, C_m} = \alpha_l d(C_k, C_l) + \alpha_m d(C_k, C_m) + \beta d(C_k, C_l) + \gamma |d(C_k, C_m) - d(C_l, C_m)| \qquad (4.1)$$

where $d_{C_k \bigcup C_l, C_m}$ is the distance between a cluster C_k and the merging of two groups C_l and C_m. The parameters are constrained such that $\alpha_l + \alpha_m + \beta = 1$, $\alpha_l = \alpha_m$, $\beta < 1$ and $\gamma = 0$.

Using these schema, a number of established agglomerative strategies can be expressed as follows:

Method	R call	α_k	β	γ
Single link (nearest neighbour)	`method = "single"`	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link (furthest neighbour)	`method = "complete"`	$\frac{1}{2}$	0	$\frac{1}{2}$
Group average link	`method = "average"`	$N_l\|(N_l+N_m)$	0	0
Weighted average link	`method = "mcquitty"`	$\frac{1}{2}$	0	0
Centroid	`method = "centroid"`	$N_l\|(N_l+N_m)$	$-N_lN_m\|(N_l+N_m)^2$	0
Incremental sum of squares	`method = "ward"`	$\frac{N_k+N_m}{N_k+N_l+N_m}$	$\frac{N_k+N_l}{N_k+N_l+N_m}$	0
Median	`method = "median"`	$\frac{1}{2}$	$-\frac{1}{4}$	0

where N_k, N_l and N_m are the cluster sizes when C_k is joined to the other two clusters considered.

This formula facilitates the use of a number of clustering approaches. Ward (1963) proposed a method in which clustering proceeds by selecting those merges which minimise the error sum of squares. If we consider the cluster specific error sum of squares:

$$ESS_k = \sum_{i+1}^{n_k}\sum_{j=1}^{p}(x_{ki,j} - \bar{\boldsymbol{x}}_{k,j})^2$$

where $\bar{\boldsymbol{x}}_{k,j}$ is the mean of cluster k with respect to variable j and $x_{ki,j}$ is the value of j for each object i in cluster k. The total error sum of squares is therefore given by $\sum_{k=1}^{K} ESS_k$ for all clusters k. This method tends to give spherical clusters, whether that is a reasonable solution to find or not.

Centroid clustering involves merging clusters with the most similar mean vectors; there is a subtle variation on a theme in which the centroid calculation is weighted. Whilst the calculations underlying these methods tend to use Euclidean distance (to facilitate interpretation in terms of the raw data) that is not compulsory. There is therefore some quite unsubtle interaction between choice of distance measure and choice of clustering algorithm which provides vast room for ambiguity in conducting cluster analysis.

4.1.5 Problems with hierarchical cluster analysis

A number of problems have been recognised with hierarchical cluster analysis. Single, complete and average linkage as well as Ward's method have the potential for reversals in the dendrogram; single and complete linkage impose a monotonicity requirement. One particular problem with single link clustering is "chaining", we could illustrate this as follows:

```
x <- rbind(mvrnorm(30, c(0,0), sigma),
  mvrnorm(30, c(8,8), sigma),
  cbind(seq(0,8, by = 0.4), seq(0,8, by = 0.4) )
dot <- hclust(dist(x), method = "single")
par(mfrow = c(1,2))
plot(dot)
plot(x, pch = cutree(dot, k = 2))
```

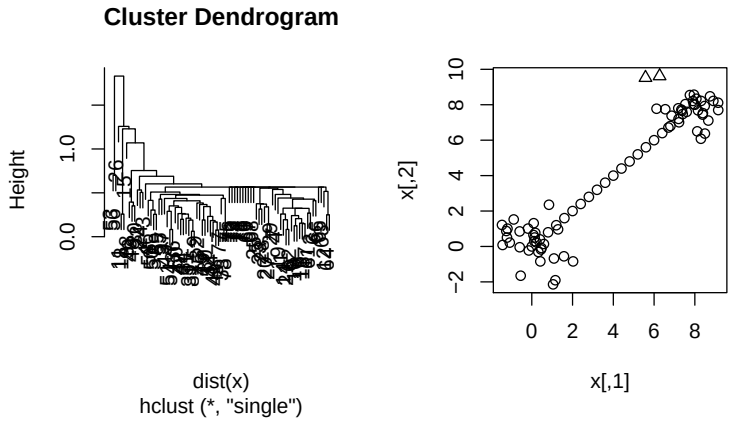


Figure 4.4: Demonstration of "chaining" with single link clustering

4.1.6 Hierarchical clustering in R

A typical example of a cluster analysis was reported by Jolliffe et al. (1986). To carry out hierarchical cluster analysis in **R** , create a distance object from the data (`USArrests.dist`), create a hierarchical clustering object from the distance matrix (`USArrests.hclust`), and plot the results. Here we have used manhattan distance and compete linkage. You may like to see how much difference the alternatives make.

```
> USArrests.dist <- dist(USArrests, method = "manhattan")
> USArrests.hclust <- hclust(USArrests.dist, method = "complete")
> plot(USArrests.hclust)
```

For comparison with other results, we may want to "cut" the dendrogram at a point which gives us a certain number of classes. Here, we will cut the dendrogram fairly high up, and look at a possible five group structure. In the following code, we use cutree to find the five groups, and produce draftsman plot with colours / symbols altering according to which of the five groups we think the state may belong to.

```
> hc.class <- cutree(USArrests.hclust, k =5)
> plot(USArrests, col = hc.class, pch = hc.class)
```

An agreement is where both objects have a one in each position. b and c denote the two possibly disagreements.

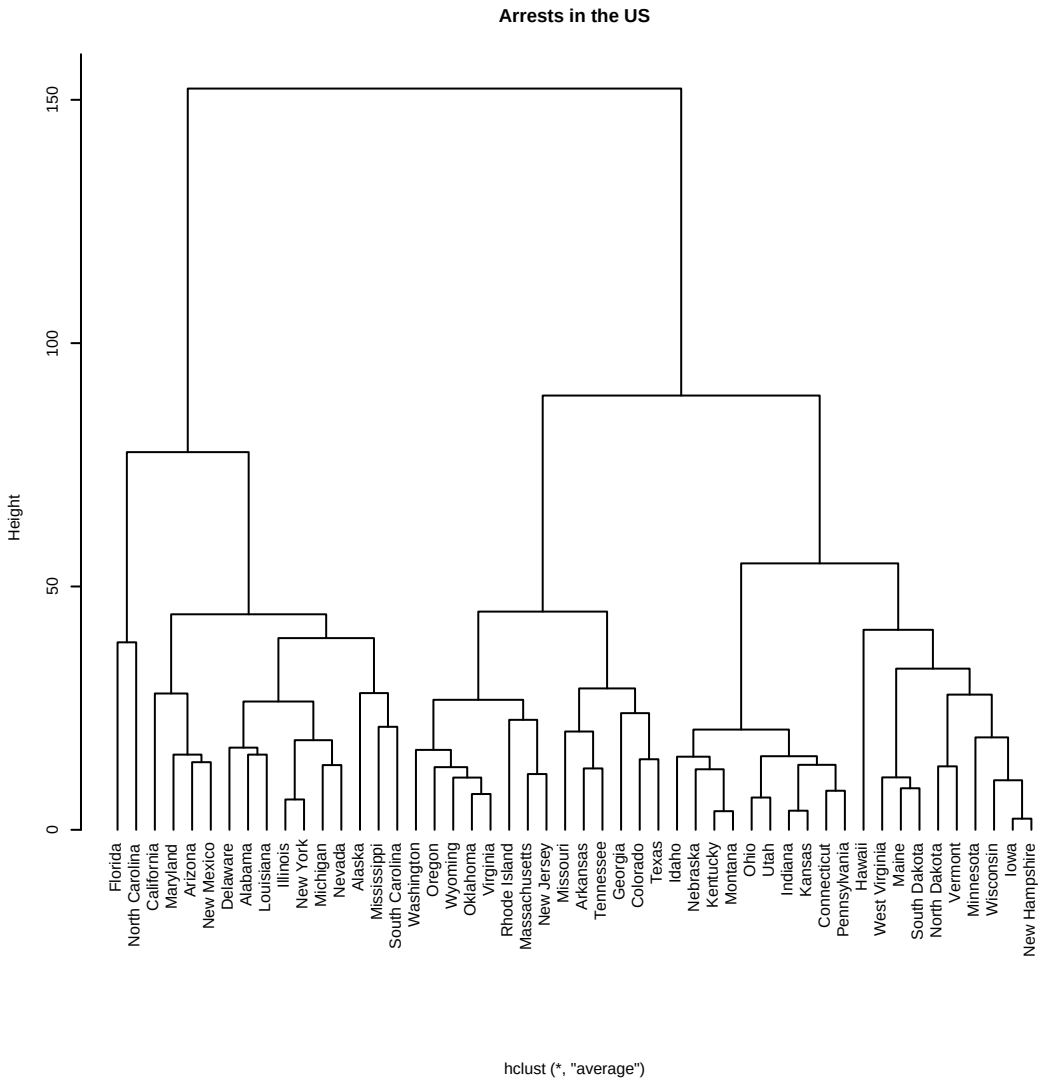


Figure 4.5: Dendrogram following complete linkage cluster analysis of US Arrests data

4.2 Cophenetic Correlation

The cophenetic correlation can be used as some kind of measure of the goodness of fit of a particular dendrogram.

$$\rho_{Cophenetic} = \frac{\sum_{i=1,j=1,i<j}^{n}(d_{ij} - \bar{d})(h_{ij} - \bar{h})}{\left(\sum_{i=1,j=1,i<j}^{n}(d_{ij} - \bar{d})^2(h_{ij} - \bar{h})^2\right)^{0.5}} \tag{4.2}$$

```
> d1 <- dist(USArrests)
> h1 <- hclust(d1)
> d2 <- cophenetic(h1)
> cor(d1, d2)
[1] 0.7636926
> h1 <- hclust(d1, method = "single")
> d2 <- cophenetic(h1)
> cor(d1, d2)
[1] 0.5702505
```

So it is easy to obtain a measure of the cophenetic correlation, it is less clear what it means. Certainly a value below 0.6 implies that there has been some distortion in the dendrogram.

4.3 Divisive hierarchical clustering

Divisive clustering reverses the approach taken above. Here, we start with one large cluster of all n objects, and split until each object is unique. (Gordon, 1999, page 90) argues that divisive clustering is not likely to lead to optimal divisions in the data. Arguably the more succesful methods are monothetic and split on one variable at each stage (see `mona()` for an R example which works with binary data).

Macnaughton-Smith et al. (1964) proposed one divisive method which has seen some use. Kaufman and Rousseeuw (1989) liken this to splintering within a political party, where one person leaves and attracts the most like-minded people and have provided the `diana()` routine which facilitates this form of clustering.

To identify the "splinter", the object with the largest average dissmilarity to all other objects is selected. Then the dissimilarities are recalculated, and any objects who are closer to the splinter than to their original group are reclassified. Average dissimilaries can then be recalculated.

At each subsequent step of the algorithm, the cluster C with the largest diameter is selected:

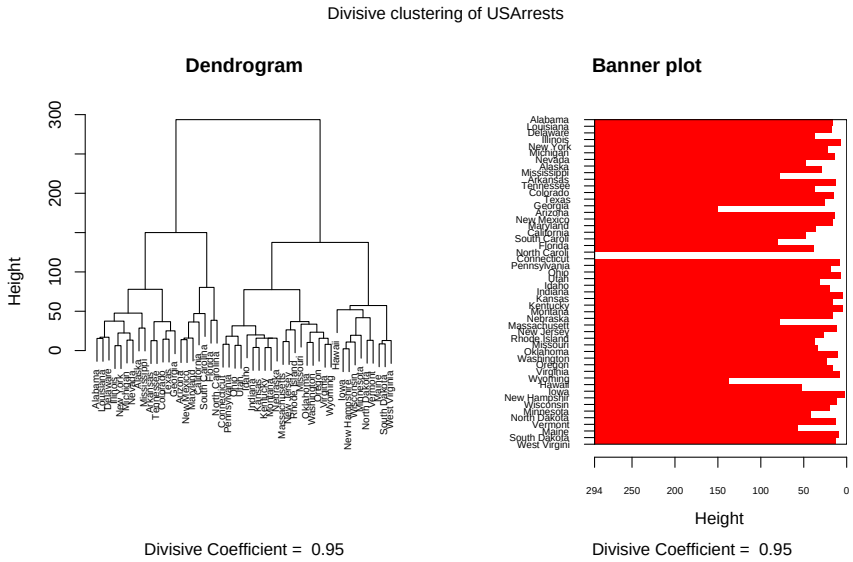


Figure 4.6: Divisive clustering of USArrests data, dendrogram and bannerplot

$$diam(C) = max_{i,j \in C} d(i,j)$$

This diameter is represented as the "height" in the dendrogram and the banner plot.

Then C is split into two subclusters, initially $A = C$ and $B = \emptyset$. For each objects in A, calculate the average dissimilarity to all other objects in A, the object with the largest distance is moved to B. This step is repeated until there are n clusters.

```
USArrests.dist <- dist(USArrests)
library(cluster)
USArrests.diana <- diana(USArrests.dist)
par(mfrow = c(1,2), oma = c(0,0,2,0))
plot(USArrests.diana, which = 2, cex = 0.6,
main = "Dendrogram", xlab = "")
plot(USArrests.diana, which = 1, main = "Banner plot",
nmax.lab = 50, cex.axis = 0.6, max.strlen = 12)
mtext("Divisive clustering of USArrests", outer = TRUE)
```

4.4 K-means clustering

This is a rather different method of clustering, aimed at finding "more homogenous" subgroups within the data. We specify at the start how many clusters we are looking for, and ideally provide some clue

as to what might be in those clusters.

The technique was perhaps first proposed by Lloyd (1957) and somewhat codified by Forgy (1965), early developments were also reported by MacQueen (1967), although the default S-Plus algorithm was developed by Hartigan and Wong (1979). There is some confusion as to whether *k-means* refers to a particular technique or a particular algorithm.

Given a number of k starting points, the data are classified, the centroids recalculated and the process iterates until stable.

We could attempt to demonstrate this with the following function

```
step <- function(X, mu1, mu2){
## classify according to current seed point (expectation step)
one <- sqrt(rowSums((X - t(t(t(mu1)) %*% t(ones)))^2))
two <- sqrt(rowSums((X - t(t(t(mu2)) %*% t(ones)))^2))
plot(x,y, col = 1 + as.numeric(one < two), pch = 16, xlim = xlims, ylim = ylims )
legend("topright", pch = c(16,16,2,3), col = c("red", "black"), legend = c("Group1", "Group2"
points(rbind(seed$mu1, seed$mu2), pch = c(2,3), col = c("red", "black"))
fixed <- (mu1 + mu2)/2
slope <- -(mu1[1] - mu2[1])/(mu1[2] - mu2[2])
abline(c(fixed[2] - slope * fixed[1], slope))

## calculate new seed points (maximisation step)
mu1 <- colMeans(X[one < two,])
mu2 <- colMeans(X[one >= two,])
return(seed = list(mu1 = mu1, mu2 = mu2))
}
```

And set up with:

```
## simulate two clusters of data
x <- c(rnorm(20,0,1), rnorm(20,4,1))
y <- c(rnorm(20,0,1), rnorm(20,4,1))
X <- cbind(x,y)
ones <- matrix(1, dim(X)[1],1)

## set up some parameters for plotting
xlims <- c(min(x), max(x)) * 1.3
ylims <- c(min(y), max(y)) * 1.3

## plot the data
par(mfrow = c(2,2))
plot(X, xlim = xlims, ylim = ylims)

## And add some very silly seed points
mu1 <- c(0,6)
mu2 <- c(5,-2)
```

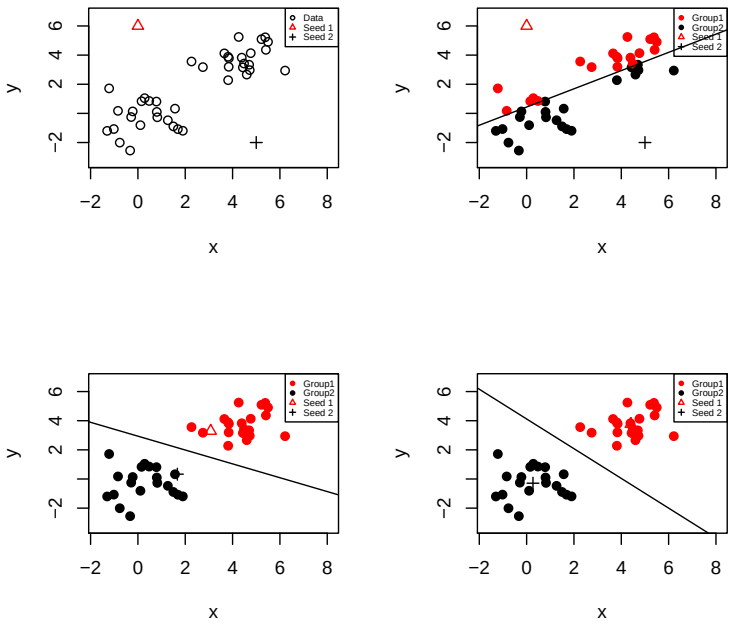


Figure 4.7: Demonstration of kmeans algorithm, points are classified according to seed, then position of seed is recalculated

```
seed <- list(mu1 = mu1, mu2 = mu2)
points(rbind(seed$mu1, seed$mu2), pch = c(2,3),col = c("red", "black"))
legend("topright", pch = c(1,2,3), col = c("black", "red", "black"), legend = c("Data", "Seed 1", "Seed
##mtext(paste("Seed points: \n Group 1 ", formatC(seed[[1]],2), "Group 2 ", formatC(seed[[2]],2)))

seed <- step(X, seed$mu1, seed$mu2)
seed <- step(X, seed$mu1, seed$mu2)
seed <- step(X, seed$mu1, seed$mu2)
```

Having illustrated the basic principles it is quite easy to run the analysis:

```
> US.km <- kmeans(USArrests, centers = 2)
> plot(USArrests, col = US.km$cluster, pch = US.km$cluster) ## not shown
> plot(prcomp(USArrests, center = TRUE)$x[,c(1,2)],
col = US.km$cluster, pch = US.km$cluster)
```

For interest, we will compare the k-means solution with the `diana()` classification:

```
kmclass <- as.vector(US.km$cluster)
```

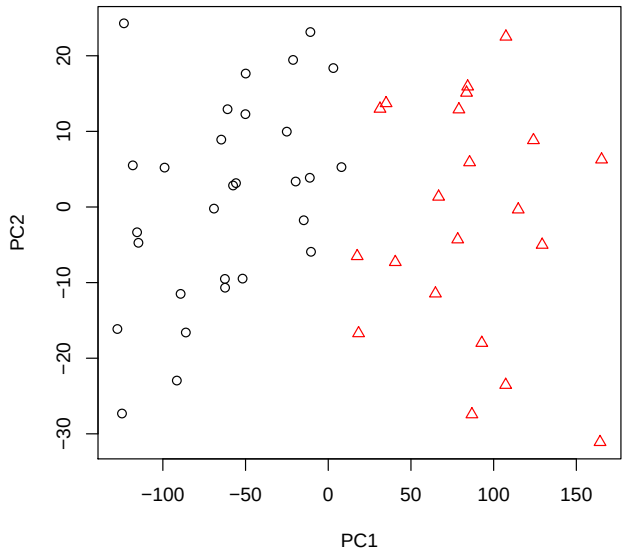


Figure 4.8: Scatterplot of original variables denoting cluster membership for $k = 2$ means clustering

```
> diana.class <- cutree(USArrests.diana, k = 2)
> xtabs(~kmclass + diana.class)
       diana.class
kmclass  1  2
      1  0 29
      2 21  0
```

So in this particular example, there is good agreement that there may be two clusters in the data.

4.4.1 Partitioning around medoids

This approach to clustering is laid out in Kaufman and Rousseeuw (1989), and the S-PLUS implementation is very well described in Struyf et al. (1997). The procedure is to find k *representative objects* called medoids in such a way that the total dissimilarity of all objects to their nearest medoids is minimised. One side-effect of this approach is that a data entity is identifed as a *representative object* which may facilitate description rather better than a non-existing group centroid. In other words, we identify objects that "best" represent their groups. This is quite simple to do in **R** , we can supply either the data matrix or a distance matrix to `pam()`, below we supply the data:

```
> USArrests.pam <- pam(USArrests, k = 2)
> par(mfrow = c(1,2))
> plot(USArrests.pam)
```

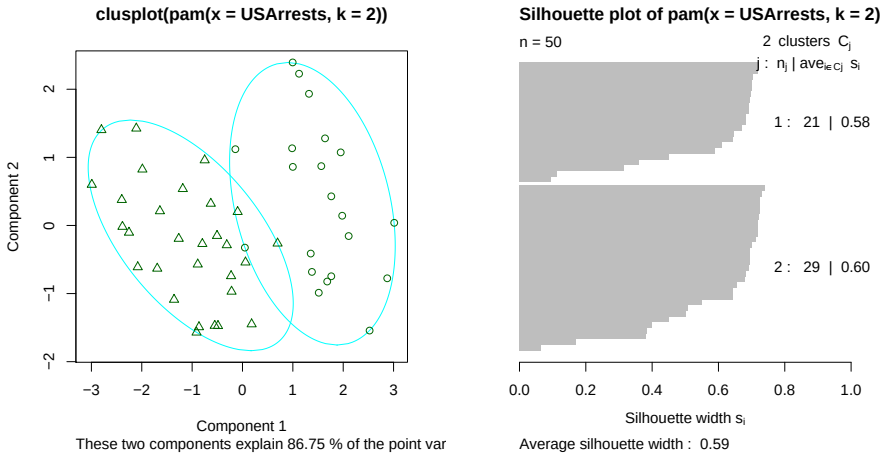


Figure 4.9: Partitioning around medoids

```
> USArrest.pam$medoids
         Murder Assault UrbanPop Rape
Michigan   12.1     255       74 35.1
Kansas      6.0     115       66 18.0
```

and it can be seen that Michigan and Kansas are in some sense the best representative examples of each of the two groups we have chosen to investigate. The `plot()` method applied to pam objects produces a 2×2 scatterplot of the first two principal components identifying group memberships and superimposing ellipses as well as a silhouette.

With minor modification, this is a tractable algorithm for large datasets. A wrapper, `clara()` has been written which makes partitioning around mediods available for a wide range of datasets.

4.4.2 Hybrid Algorithms

There have been a number of proposals in the literature recently for hybrid algrithms. We consider HOPACH, essentially it is a divisive algorithm, but at each stage a merging step is incorporated to bring together similar clusters (which may be on different branches of the tree). Again, this method produces medoids, in this case 13 were identified.

```
> library(hopach)
> USArrests.hopach <- hopach(USArrests)
> row.names(USArrests)[USArrests.hopach$clustering$medoids]
 [1] "Mississippi"   "Alaska"        "New Mexico"    "Michigan"
 [5] "California"    "Illinois"      "Missouri"      "West Virginia"
```

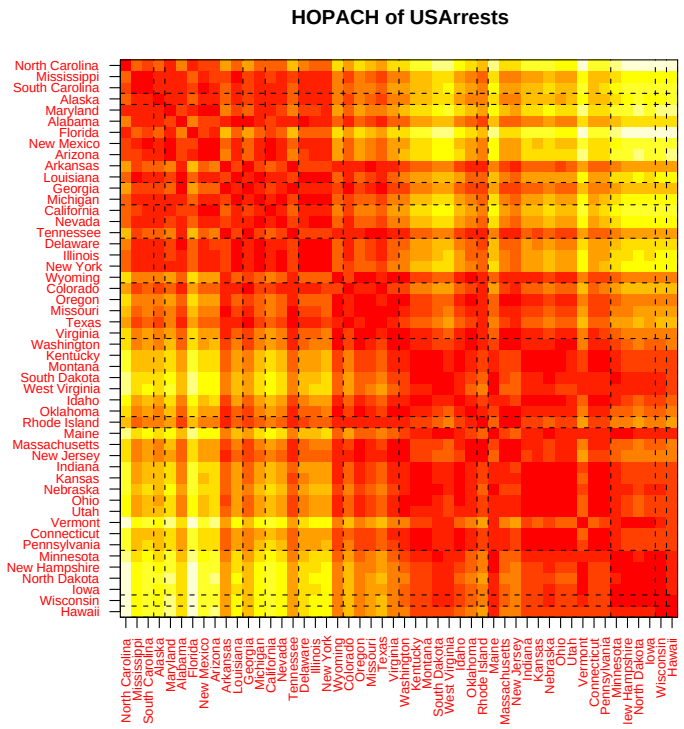


Figure 4.10: Heatmap of distances organised according to output of HOPACH with cluster groups denoted by dashed lines

```
 [9] "Rhode Island"  "Nebraska"      "New Hampshire" "Wisconsin"
[13] "Hawaii"
> dplot(dist(USArrests), us, labels = row.names(USArrests), main = "HOPACH of USArrests")
```

The `dplot()` method requires input of a distnce matrix and produces a heatmap of the distances ordered by the cluster structure. Silhouette plots are also available, and further plot methodology is available from programs external to **R** .

4.5 K-centroids

More recent proposals in the literature have tried to remove the dependency on algorithmic dependency. I need to add something on this shortly.

4.6 Further information

It is worth repeating that "cluster analysis" is a poorly delimited set of methods for unsupervised classification. Obvious omissions include fuzzy analysis (where we accept some uncertainty in group assignments), monothetic analysis (where we split based on binary variables one at a time). All these merit further consideration. However, it is acknowledged that the greatest omission at present is in terms of mixture modelling, much more comprehensive information is available in McLachlan and Peel (2000), some coverage is also given in Flury (1997).

Chapter 5

Multidimensional scaling

We have looked at clustering techniques which partition an $n \times n$ matrix of dissimilarities into groups of like individuals. Whilst it is possible to visualise differences between individuals in terms of dendrograms, it might be more useful to have a direct representation of the relationship between individuals. We are now going to consider techniques which let us visualise the relative distances bewteen individuals in a low dimensional structure. Some early ideas were set out by Richardson (1938), and the algebraic techniques required to reconstruct a configuration from distances were established by Young and Householder (1938). Torgerson (1952) set out the foundations for this work, but developments in the technique associated with the name *principal co-ordinates analysis* were given by Gower (1966). Arguably, the tecnique is most commonly referred to as *scaling*, often as classical scaling.

5.1 Metric Scaling

Consider (any) distance matrix $\mathbf{\Delta}$. It is metric if elements of $\mathbf{\Delta}$ satisfy the metric (triangle) inequality $\delta_{ij} \leq \delta_{ik} + \delta_{kj}$ for all i, j, k.

Classical metric multidimensional scaling is based on the $n \times n$ distance or dissimilarity matrix, we will note later some connections with principal components. Here, we will first consider a matrix $\mathbf{\Delta}$ of *Euclidean* distances between objects, such that δ_{ij} is the distance between points i and j. These n points can be represented in $n-1$ dimensional space. We are looking for a configuration of n points such that the distance d_{ij} between points i and j equals the dissimilarity δ_{ij} for all i, j. The dimensionality of this configuration is q such that we are seeking to reconstruct an $n \times q$ matrix $\boldsymbol{X}$.

We can very easily get from an $n \times p$ matrix $\boldsymbol{X}$ to a Euclidean distance matrix. If we first form the $n \times n$ matrix $\boldsymbol{Q}$ then $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^T$. Considering this one element at a time:

$$q_{rs} = \sum_{j=1}^{p} x_{rj} x_{sj} \tag{5.1}$$

We know that the Euclidean distance is given by:

$$\begin{aligned} \delta_{rs}^2 &= \sum_{j=1}^{p} (x_{rj} - x_{sj})^2 \\ &= \sum_{j=1}^{p} (x_{rj}^2 + \sum_{j=1}^{p} (x_{sj}^2 - 2 \sum_{j=1}^{p} x_{rj} x_{sj} \\ &= q_{rr} + q_{ss} - 2q_{rs} \end{aligned}$$

So given $\boldsymbol{X}$ we can find $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^T$ and hence find the Euclidean distance.

What we want to do now is reverse this process. We should note that our recovered $n \times q$ matrix is not uniquely defined - it is only defined up to translation, rotation and reflection. To fix this, we will usually assume $\boldsymbol{X}$ has been centred (i.e. make column means zero, such that $\sum_{i=1}^{n} y_{ij} = 0$). It may be noted here that we don't necessarily want to recover an $n \times p$ matrix, we can reconstruct a matrix with up to $n - 1$ columns, but as with other dimension reduction techniques we hope to find an $n \times q$ matrix with $q < p$; clearly if $q = 2$ it will be easy to visualise the results. So, to recover $\boldsymbol{Q}$ from $\boldsymbol{D}$:

$$\sum_{r=1}^{n} d_{rs}^2 = trace(\boldsymbol{Q}) + nq_{ss} \tag{5.2}$$

$$\sum_{s=1}^{n} d_{rs}^2 = nq_{rr} + trace(\boldsymbol{Q}) \tag{5.3}$$

$$\sum_{r=1}^{n} d_{rs}^2 \sum_{s=1}^{n} d_{rs}^2 = 2ntrace(\boldsymbol{Q}) \tag{5.4}$$

By rearranging this and manipulating the equations above which lead us to our distance matrix, we can recover elements of $\boldsymbol{Q}$ from $\boldsymbol{D}$ using a double centering procedure:

$$q_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{i\cdot}^2 - d_{\cdot j}^2 + d_{\cdot\cdot}^2)$$

The dots denote means taken over the relevant indices.

In summary, to find $\boldsymbol{Q}$ given $\boldsymbol{D}$:

- Square it element by element
- Double centre it
  - subtract column means
  - subtract row means
  - add overall mean
- Multiply by $-\frac{1}{2}$.

Having found $\boldsymbol{Q}$ ($\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^T$), all we need is to find a suitable $\boldsymbol{X}$, which sounds like some kind of matrix square root problem. Given Euclidean distances, $\boldsymbol{Q}$ is symmetric and we can do this using the spectral decomposition $\boldsymbol{Q} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^T$ where $\boldsymbol{\Lambda} = \lambda_1, \lambda_2, \ldots, \lambda_n$, a diagnonal matrix of ordered eigenvalues and $\boldsymbol{E}$ is the matrix whose columns are the corresponding (normalised) eigenvectors. If $\boldsymbol{Q} = \boldsymbol{E}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{E}^T = \boldsymbol{X}\boldsymbol{X}^T$ then $\boldsymbol{X} = \boldsymbol{E}\boldsymbol{\Lambda}^{\frac{1}{2}}$ and we have recovered the co-ordinates from the inter-point distances.

$$\boldsymbol{X} = \left(\sqrt{\lambda_1} \begin{pmatrix} e_{11} \\ e_{12} \\ \vdots \\ e_{1n} \end{pmatrix}, \ldots, \sqrt{\lambda_n} \begin{pmatrix} e_{n1} \\ e_{n2} \\ \vdots \\ e_{nn} \end{pmatrix} \right)$$

So if we want a one dimensional representation, we just use $\left(\sqrt{\lambda_1}\boldsymbol{e}_1\right)$, for a two dimensional representation we would use $\left(\sqrt{\lambda_1}\boldsymbol{e}_1, \sqrt{\lambda_2}\boldsymbol{e}_2\right)$

5.1.1 Similarities with principal components analysis

A short diversion noting a few similarities with principal component analysis may be in order. Consider the centred data matrix $\boldsymbol{X}$:

$$C = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X}$$

Principal components come from an eigenanalysis of $\boldsymbol{C}$, here we denote the eigenvalues of $\boldsymbol{C}$ by μ_i and associated eigenvectors by $\boldsymbol{a}_i$:

$$\begin{aligned}
\boldsymbol{C}\boldsymbol{a}_i &= \mu_i \boldsymbol{a}_i \\
\frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{a}_i &= \mu_i \boldsymbol{a}_i \\
\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{a}_i &= (n-1)\mu_i \boldsymbol{a}_i \\
\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{a}_i &= (n-1)\mu_i \boldsymbol{X}\boldsymbol{a}_i \\
\boldsymbol{Q}\underbrace{\boldsymbol{X}\boldsymbol{a}_i}_{\boldsymbol{z}_i} &= (n-1)\mu_i \underbrace{\boldsymbol{X}\boldsymbol{a}_i}_{\boldsymbol{z}_i}
\end{aligned}$$

So $\boldsymbol{X}\boldsymbol{a}_i = \boldsymbol{z}_i$ is an eigenvector of $\boldsymbol{Q}$ with corresponding eigenvalue $(n-1)\mu_i$. If we want a normalised eigenvector it may be worth noting that the length can be found as follows:

$$\begin{aligned}
||\boldsymbol{z}_i||^2 = \boldsymbol{z}_i^T\boldsymbol{z}_i = \boldsymbol{a}_i\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{a}_i &= (n-1)\boldsymbol{a}_i^T\boldsymbol{C}\boldsymbol{a}_i \\
&= (n-1)\boldsymbol{a}_i^T\mu_i\boldsymbol{a}_i \\
&= (n-1)\mu_i\frac{\boldsymbol{a}_i^T\boldsymbol{a}_i}{||\boldsymbol{a}_i||^2} \\
&= (n-1)\mu_i
\end{aligned}$$

So, $||\boldsymbol{z}_i|| = \sqrt{(n-1)\mu_i}$. Hence a normalised eigenvector for $\boldsymbol{Q}$ takes the form $\frac{1}{(n-1)\mu_i}\boldsymbol{X}\boldsymbol{a}_i$ with eigenvalue $(n-1)\mu_i$

Therefore, our eigenvalues and eigenvectors found from multidimensional scaling / principal co-ordinates analysis are related to those found from decomposition of the covariance of the scaled data matrix:

$$\begin{aligned}
\lambda_i &= (n-1)\mu_i \\
e_i &= \frac{1}{\sqrt{\lambda_i}}\boldsymbol{X}\boldsymbol{a}_i
\end{aligned}$$

Remember that $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{A}^T$, where:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_n^T \end{pmatrix}$$

So $\boldsymbol{X}\boldsymbol{a}_i$:

$$(\boldsymbol{X}\boldsymbol{a}_1\boldsymbol{X}\boldsymbol{a}_2, \ldots, \boldsymbol{X}\boldsymbol{a}_n)$$

in other words, this is our matrix of principal component scores.

5.2 Visualising multivariate distance

Consider the US Arrests data (we have looked at this a few times). If you still have the distance matrix `spot` created earlier, you can run principal co-ordinates analysis quite easily:

```
> spot <- dist(USArrests, method = "euclidean")
> what <- cmdscale(spot)
> plot(what[,1], what[,2],
   xlab = "Axis 1", ylab = "Axis 2",
   main = "US Arrests")
> identify(what[,1], what[,2], row.names(USArrests))
```

By default you extract two variables, and you don't get the eigen values. You can alter the function call if you want to change that (see `?cmdscale`)

You might like to use `identify()` to check that very odd points according to principal co-ordinates are also very odd according to your cluster analysis.

5.3 Assessing the quality of fit

One measure of discrepancy is given by:

$$\varphi = \sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij}^2 - d_{ij}^2)$$

and it can be shown (Mardia et al., 1979)that:

$$\varphi = 2n(\lambda_{q+1} + \ldots + \lambda_n)$$

So, if we fit a model with $q = n - 1 = 49$ (in order to measure the size of the discarded eigenvalues - most of these eigenvalues are zero hence warnings about eigenvalues below zero):

```
> what <- cmdscale(spot, eig = TRUE, k = 49)
> 2 * dim(USArrests)[1] * sum(what$eig[3:49])
```

We should find that this give the same value as a direct comparison of the distance matrices formed from the data and the $q = 2$ dimensional representation. (Note that we convert the distance matrices into ordinary matrices, and then we carry out vectorised operations to take squares, subtract elementwise and sum)

```
what <- cmdscale(spot, eig = TRUE, k = 2)
delta <- as.matrix(dist(USArrests, upper = TRUE))
d <- as.matrix(dist(what$points, upper = TRUE))
sum(as.vector(delta)^2 - as.vector(d)^2)
```

This has clear analogies with the percent trace measure used in principal components analysis:

$$\frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{5.5}$$

```
> what <- cmdscale(spot, eig = TRUE, k = 4)
> what$eig / sum(what$eig)
[1] 0.9655342206 0.0278173366 0.0057995349 0.0008489079
```

Considerable work has been carried out on Goodness of fit measures for scaling problems. If $\mathbf{\Delta}$ is based on a measure other than the Euclidean then our reconstructed $\boldsymbol{Q}$ may not be positive semi-definite (in other words we find some negative eigenvalues and imaginary co-ordinates). If $\boldsymbol{Q} = \sum_{i=1}^{q} \lambda_i \boldsymbol{e}_i \boldsymbol{e}_i^T$ then Mardia (1978) suggests the discrepancy measure:

$$\varphi\prime = trace(\boldsymbol{Q} - \hat{\boldsymbol{Q}}^2) \tag{5.6}$$

. Following Eckart and Young (1936) we would use:

$$\frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} |\lambda_i|} \tag{5.7}$$

or following Mardia (1978) we would use:

$$\frac{\sum_{i=1}^{q} \lambda_i^2}{\sum_{i=1}^{p} \lambda_i^2} \tag{5.8}$$

5.3.1 Sammon Mapping

Classical metrical scaling works on orthogonal projections, and has the attractive property of an exact analytical solution. This is quite restrictive, Sammon (1969) suggested minimising the discrepancy measure:

$$\varphi\prime\prime = \sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij} - d_{ij})^2$$

This has no analytical solution, and numerical methods must be used. But it is worth noting that a set of disparities are generated:

$$\hat{d}_{ij} = a + b\delta_{ij} \tag{5.9}$$

This should look like a reasonably familiar formula, and you should have guessed that residual sums of squares from 5.9 yields another discrepancy measure. This measure can (should / must) be normalised with reference to its size $\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2$, giving what is called the **st**andardised **re**sidual **s**um of **s**quares:

$$STRESS = \left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2}\right)^{\frac{1}{2}} \tag{5.10}$$

$$SSTRESS = \left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}^2 - \hat{d}_{ij}^2)^2}{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^4}\right)^{\frac{1}{2}} \tag{5.11}$$

Normalisation means that both these measures take on values between 0 and 1, lower values indicate better fit. Values below 0.1 are usually considered adequate, but it may be worth noting that Kruskal (1964) suggests values below 0.2 give a poor fit, values below 0.1 a fair fit, values below 0.05 a good fit, values below 0.025 an excellent fit.

Chapter 6

Multivariate normality

The multivariate normal distribution remains central to most work concerning multivariate continuous data. Experience has suggested that it is usually at least an acceptable approximation, and of course one usually has recourse to the central limit theorem. Although we will examine a few common alternatives, it is perhaps in the field of robust statistics where it's use has been modified most.

6.1 Expectations and moments of continuous random functions

The mean and covariance can be defined in a similar way to to the univariate context

Definition 6.2 *Given a multivariate distribution function for p random variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ taking the form $P(\boldsymbol{x} \in A) = \int_A f(\boldsymbol{x})d\boldsymbol{x}$, expectation can be defined as:*

$$E(g(\boldsymbol{x})) = \int_{\mathscr{R}} g(\boldsymbol{x})f(\boldsymbol{x})d\boldsymbol{x} \tag{6.1}$$

which gives rise to moments

$$E(\boldsymbol{x}) = \boldsymbol{\mu} \tag{6.2}$$

as well as moment generating functions:

$$M_x\boldsymbol{t} = Ee^{\boldsymbol{x}^T\boldsymbol{t}}, \tag{6.3}$$

cumulants:

$$K_x\boldsymbol{t} = \log(M_x\boldsymbol{t}), \tag{6.4}$$

and the characteristic function:

$$\phi_x\boldsymbol{t} = E(Ee^{i\boldsymbol{x}^T\boldsymbol{t}}) \tag{6.5}$$

These generating functions have analogous properties to their univariate counterparts.

6.3 Multivariate normality

Definition 6.4 *If $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ is a p dimensional vector of random variables, then $\boldsymbol{y}$ has a multivariate normal distribution if its density function is:*

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})};$$

for $-\infty < y_{ij} < \infty, j = 1, 2, \ldots, p$

And it can be shown that $E(\boldsymbol{x}) = \boldsymbol{\mu}$, and that $Var(\boldsymbol{x}) = \boldsymbol{\Sigma}$, hence we can use the notation:

$$\boldsymbol{y} \sim MVN_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Finding the maximum likelihood estimators for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is not trivial, there are perhaps at least three derivations. We briefly recap results from one of the more popular derivations here.

Theorem 6.5 *If $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ is a p dimensional vector of random variables representing a sample from $MVN_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the log-likelihood function can be given by as follows:*

$$(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}) = -\frac{np}{2}\log(2\pi) - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{i=1}^{n}\left((\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu})\right) \tag{6.6}$$

which can be rewritten as:

$$(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}) = -\frac{np}{2}\log(2\pi) - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}trace\left(\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})^T(\boldsymbol{x}_i - \boldsymbol{\mu})\right)$$

adding and subtracting $\bar{\boldsymbol{x}}$ from each of the two brackets on the right, and setting $\boldsymbol{A} = \sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$ allows this to be rewritten as:

$$-\frac{np}{2}\log(2\pi) - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}trace\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{A} + n\boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{x}}-\boldsymbol{\mu})(\bar{\boldsymbol{x}}-\boldsymbol{\mu})^T\right)$$

The way to obtaining the maximum likelihood estimators for $\boldsymbol{\mu}$ is now fairly clear. As $\boldsymbol{\Sigma}$ is positive definite, we require $(\bar{\boldsymbol{x}}-\boldsymbol{\mu})(\bar{\boldsymbol{x}}-\boldsymbol{\mu})^T$ to be greater than or equal to zero, hence $\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{x}}$ maximises the likelihood for all positive definite $\boldsymbol{\Sigma}$

A number of derivations for Sigma are possible, essentially we need to minimise:

$$(\bar{\boldsymbol{x}}, \boldsymbol{\Sigma}) = \log|\boldsymbol{\Sigma}| + trace(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}). \tag{6.7}$$

with respect to $\boldsymbol{\Sigma}$. This can be derived as a minimisation of $(\bar{\boldsymbol{x}}, \boldsymbol{\Sigma}) - \log(|\boldsymbol{S}|) = trace(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) - \log|\boldsymbol{\Sigma}^{-1}\boldsymbol{S}|$. If $\boldsymbol{S}^{1/2}$ is the positive definite symmetric square root of $\boldsymbol{S}$, $trace(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) = trace(\boldsymbol{S}^{1/2}\boldsymbol{\Sigma}^{-1}\boldsymbol{S}^{1/2})$, but given that $\boldsymbol{A} = \boldsymbol{S}^{1/2}\boldsymbol{\Sigma}^{-1}\boldsymbol{S}^{1/2}$ is a symmetric matrix we know that $trace(\boldsymbol{A}) = \sum_{j=1}^{p}\lambda_i$ and $|\boldsymbol{A}| = \prod_{j=1}^{p}\lambda_i$ where $\lambda_1, \ldots, \lambda_p$ are the eigenvalues of $\boldsymbol{A}$, which must all be positive (because $\boldsymbol{A}$ is positive definite). Hence we wish to minimise:

$$(\bar{\boldsymbol{x}}, \boldsymbol{\Sigma}) - \log(|\boldsymbol{S}|) = \sum_{j=1}^{p}\lambda_i - \log\prod_{j=1}^{p}\lambda_i \tag{6.8}$$

and given that $f(z) = z - \log(z)$ takes a unique mininum at $z = 1$ we wish to find a matrix where all the eigenvalues equal 1, i.e. the identity matrix. Consequently we wish to find:

$$\boldsymbol{S}^{1/2}\boldsymbol{\Sigma}^{-1}\boldsymbol{S}^{1/2} = \boldsymbol{I} \tag{6.9}$$

hence $\boldsymbol{S}$ is the maximum likelihood estimator of $\boldsymbol{\Sigma}$. Do note that this requires $n > p$.

6.5.1 R estimation

`cov()` and `var()` (equivalent calls) both give the unbiased estimate for the variance-covariance matrix, i.e. $\frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$. It is worth nothing that by default, `cov.wt()` (which will do a few other things as well) uses the divisor $\frac{1}{n}$

6.6 Transformations

Box and Cox (1964) modified earlier proposals by Tukey (1957) to yield the following transformation:

$$x^{(\lambda)} = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \lambda \neq 0, \\ \log x, & \lambda = 0 \text{ and} x > 0. \end{cases}$$

$$\mathscr{L}(\mu, \sigma^2, \lambda | x^{(\lambda)}) \propto (2\pi\sigma^2)^{-n/2} \exp(-\sum_{i=1}^{n} \frac{(x_i^{(\lambda)} - \mu)^2}{2\sigma^2} \prod_{i=1}^{n} x_i^{\lambda-1} \tag{6.10}$$

If λ is fixed, this likelihood is maximised at:

$$\bar{x}^{(\lambda)} = \frac{1}{n} \sum_{i=1} n x_i^{(\lambda)} \tag{6.11}$$

and

$$s^2(\lambda) = \frac{1}{n} \sum_{i=1}^{n} (x_i^{(\lambda)} - \bar{x}^{(\lambda)})^2 \tag{6.12}$$

which means that the value maximising the log-likelihood is proportional to:

$$\mathscr{L}(\lambda) = -\frac{n}{2} \log s^2(\lambda) + (\lambda - 1) \sum_{i=1}^{n} \log x_i \tag{6.13}$$

It is possible to apply this transformation to each variable in turn to obtain marginal normality Gnanadesikan (1977) argues that this can be used satisfactorily in many cases.

However, it may be preferable to carry out a multivariate optimisation of the transformation parameters.

A range of tranformations have been considered for multivariate data, mainly of the Box-Cox type. If the variables $\boldsymbol{y} = (y_1, y_2, \ldots, y_p$ are are smooth transformation of $\boldsymbol{x}$, the frequency function for $\boldsymbol{y}$ can be given by:

$$g(\boldsymbol{y}) = f(\boldsymbol{x}(\boldsymbol{y}))|\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}|$$

where $\boldsymbol{x}(\boldsymbol{y})$ is $\boldsymbol{x}$ expressed in terms of the elements of $\boldsymbol{y}$, and $J = |\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}|$ is the Jacobian which ensures the density is mapped correctly.

$\prod_{j=1}^{p} \prod_{i=1}^{n} x_{ij}^{\lambda_j - 1}$

$$\mathscr{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\lambda} | \boldsymbol{X^{(\lambda)}}) \propto -\frac{n}{2} \log|\boldsymbol{\Sigma}| - \frac{1}{2} tr(\boldsymbol{\Sigma}^{-1}(\boldsymbol{X^{(\lambda)}} - \boldsymbol{1\mu}^T)^T(\boldsymbol{X^{(\lambda)}} - \boldsymbol{1\mu}^T) + sum_{j=1}^{p} \left((\lambda_j - 1) \sum_{i=1}^{n} \log x_{ij} \right) \tag{6.14}$$

If λ is fixed, this likelihood is maximised at:

$$\bar{x}^{(\lambda)} = \frac{1}{n}\mathbf{1}^T\boldsymbol{X} \tag{6.15}$$

and

$$s^2(\lambda) = (\boldsymbol{X}^{(\boldsymbol{\lambda})} - \mathbf{1}\boldsymbol{\mu}^T)^T(\boldsymbol{X}^{(\boldsymbol{\lambda})} - \mathbf{1}\boldsymbol{\mu}^T) \tag{6.16}$$

which means that the value maximising the log-likelihood is proportional to:

$$\mathscr{L}(\lambda) = -\frac{n}{2}\log|\hat{\boldsymbol{\Sigma}}| + \sum_{j=1}^{p}\left((\lambda_j - 1)\sum_{i=1}^{n}\log x_{ij}\right) \tag{6.17}$$

Chapter 7

Inference for the mean

We introduced the Mahalanobis distance earlier in 3.1. Consideration of the *squared* Mahalabnobis distance leads us to consider the so called T^2 statistic (the nomenclature reflects that this relates to a t-statistic for one variable). This can be found as:

$$T^2 = n(\boldsymbol{\mu}_0 - \bar{\boldsymbol{x}})^T \boldsymbol{S} (\boldsymbol{\mu}_0 - \bar{\boldsymbol{x}}) \tag{7.1}$$

where n is the sample size, $\boldsymbol{\mu}_0$ is the hypothesised mean, $\bar{\boldsymbol{x}}$ and $\boldsymbol{S}$ are the sample mean and covariance matrices respectively. It turns out that this statistic follows a T^2 distribution, however, given that there is a simple relationship between the T^2 and F distribution it is often easier to work with the latter.

If $\boldsymbol{x}_i$, $i = 1, \ldots n$ represent a sample from a p variate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, provided $\boldsymbol{\Sigma}$ is positive definite and $n > p$, given sample estimators for mean and covariance $\bar{\boldsymbol{x}}$ and $\boldsymbol{S}$ respectively, then:

$$F = \left(\frac{n}{n-1}\right)\left(\frac{n-p}{p}\right)(\boldsymbol{\mu}_0 - \bar{\boldsymbol{x}})^T \boldsymbol{S} (\boldsymbol{\mu}_0 - \bar{\boldsymbol{x}}) \tag{7.2}$$

follows an F-distribution with p and $(n-p)$ degrees of freedom. Note the requirement that $n > p$, i.e. that $\boldsymbol{S}$ is non-singular. This clearly limits the use of this test in bio-informatic applications and we may examine a few proposals to deal with this. To carry out a test on $\boldsymbol{\mu}$, we determine whether $F \leq F_{(1-\alpha),p,n-p}$, the $(1-\alpha)$ quantile of the F distribution on p and $n-p$ degrees of freedom. We reject the null hypothesis if our test statistic exceeds this value. We will not consider the one sample T^2 test any further, but will now examine the two-sampled test.

7.1 Two sample Hotelling's T^2 test

Analagous to the univariate context, we wish to determine whether the mean vectors are comparable, more formally:

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 \tag{7.3}$$

The T^2 statistic proposed by Hotelling (1931), will be based this time on the distance between two mean vectors. It can be calculated as:

$$T^2 = \left(\frac{n_1 n_2}{n_1 + n_2}\right)(\bar{\boldsymbol{x}}_{\mathbf{1}} - \bar{\boldsymbol{x}}_{\mathbf{2}})^T \boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}_{\mathbf{1}} - \bar{\boldsymbol{x}}_{\mathbf{2}}) \tag{7.4}$$

where S^{-1} is the inverse of the pooled correlation matrix given by:

$$\boldsymbol{S} = \frac{(n_1 - 1)\boldsymbol{S}_{\mathbf{1}} + (n_2 - 1)\boldsymbol{S}_{\mathbf{2}}}{n_1 + n_2 - 2}$$

given the sample estimates for covariance, $\boldsymbol{S}_{\mathbf{1}}$ and $\boldsymbol{S}_{\mathbf{2}}$ in the two samples. As before, there is a simple relationship between the test statistic, T^2, and the F distribution.

If $\boldsymbol{x}_{1i}$, $i = 1, \ldots n_1$ and $\boldsymbol{x}_{2i}$, $i = 1, \ldots n_2$ represent independent samples from two p variate normal distribution with mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ but with common covariance matrix $\boldsymbol{\Sigma}$, provided $\boldsymbol{\Sigma}$ is positive definite and $n > p$, given sample estimators for mean and covariance $\bar{\boldsymbol{x}}$ and $\boldsymbol{S}$ respectively, then:

$$F = \frac{(n_1 + n_2 - p - 1)T^2}{(n_1 + n_2 - 2)p}$$

has an F distribution on p and $(n_1 + n_2 - p - 1)$ degrees of freedom. Essentially, we compute the test statistic, and see whether it falls within the $(1 - \alpha)$ quantile of the F distribution on those degrees of freedom. note again that to ensure non-singularity of $\boldsymbol{S}$, we require that $n_1 + n_2 > p$.

Whilst we won't use the next formula for computation, it may clarify understanding of this test if we consider the sample estimate of the mean difference $\boldsymbol{d} = \bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2$, and the corresponding population distance $\boldsymbol{\delta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ we can use the formula:

$$F = \left(\frac{n_1 + n_2 - p - 1}{p(n_1 + n_2 - 2)}\right)\left(\frac{n_1 n_2}{n_1 + n_2}\right)(\boldsymbol{d} - \boldsymbol{\delta})^T \boldsymbol{S}^{-1}(\boldsymbol{d} - \boldsymbol{\delta})$$

to calculate the test statistic.

We are going to consider an example using data from Flea Beetles reported by Lubischew (1962) and used in (Flury, 1997, page 307). It should be noted that in terms of practical computation, methods are based on the QR decomposition will be used, details are given in Seber (1984). However, for the purposes of understanding the principles behind the test, we follow the formula directly.

```
> library(Flury)
> ?flea.beetles
> data(flea.beetles)
```

It can be seen that there is a factor "Species" denoting whether the beetles are from 'oleracea' or 'carduorum'. There are four numeric variables as follows: 'TG'; Distange of the Transverse Groove to the posterior border of the prothorax (microns), 'Elytra'; Length of the Elytra (in units of 0.01mm), 'Second.Antenna'; Length of the second antennal joint (microns) and 'Third.Antenna'; Length of the third antennal joint (microns). We need to estimate the mean for each sample, and calculate the difference between the two vectors:

```
mu <- by(flea.beetles[,-1], flea.beetles$Species, colMeans)
mudiff <- mu[[1]] - mu[[2]]
p <- dim(flea.beetles)[2] - 1 ## how many variables are we using
```

The next step is to extract the two covariance matrices:

```
> covmats <- by(flea.beetles[,-1], flea.beetles$Species, cov)
> covmats
```

and then to estimate the pooled covariance matrix $\boldsymbol{S}$ for the flea beetle data (where N[1] gives n_1, N[2] gives n_2), can be calculated as:

```
> N <- xtabs(~flea.beetles[,1])
> pooledS <- ((N[1]-1) * covmats[[1]] + (N[2]-1) * covmats[[2]]) / (N[1] + N[2] -2)
> pooledS
> Sinv <- solve(pooledS)
> Sinv
                          TG        Elytra Second.Antenna Third.Antenna
TG              0.013257964 -0.0053492256   0.0015134494 -0.0021617878
Elytra         -0.005349226  0.0066679441  -0.0047337699 -0.0005969439
Second.Antenna  0.001513449 -0.0047337699   0.0130490933 -0.0007445297
Third.Antenna  -0.002161788 -0.0005969439  -0.0007445297  0.0060093005
```

Having calculated the inverse of the pooled correlation matrix we also need the scaling factor $\frac{n_1 n_2}{n_1+n_2}$. Hotellings T^2 is then quite straightforward to calculate:

```
> scaleFact <- (N[1]*N[2]) / (N[1]+N[2])
> Hotellings <-  t(mudiff) %*% Sinv %*% mudiff * scaleFact
> Hotellings
          [,1]
[1,] 133.4873
```

which is the value of the T^2 statistic. We could work with this value directly, but it is more convenient to transform it into something we can compare with the F distribution.

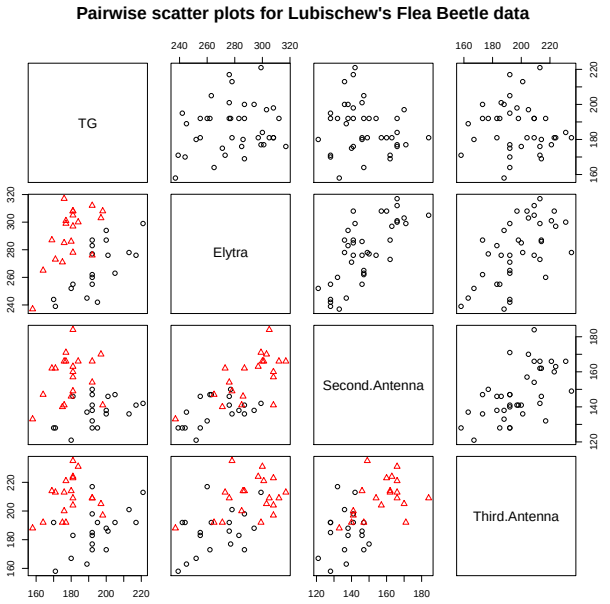


Figure 7.1: Scatterplot for Lubischew flea data, lower panel has symbols denoting the two species

```
test <- ((N[1] + N[2] - p - 1) * Hotellings )/ ((N[1] + N[2] - 2) * p)
test
       [,1]
[1,] 30.666
```

and we compare this with an F distribution having p and $(n_1 + n_2 - p - 1)$ d.f.

And we can check this as follows:

```
> pf(test, p, N[1]+N[2]-p-1,lower.tail = FALSE )
              [,1]
[1,] 3.215324e-11
```

which gives us the area under the curve from our test statistic (30.666) to ∞. Clearly in this case, we have reject H_0, i.e. there is evidence that the mean vectors, $\bar{\boldsymbol{x}}_{oleracea} = (194.4737, 267.0526, 137.3684, 185.9474), \bar{\boldsymbol{x}}_{car}$ $(179.55, 290.80, 157.20, 209.25)$, for the two species differ. This is perhaps no surprise if you consider the data. Figure 7.1 contains a scatterplot where different symbols have been used in the lower panels for the two species. However, we do need to consider this in a little more detail.

7.2 Constant Density Ellipses

Flury (1997) gives an interpretation of constant density ellipses in terms of the Mahalanobis distance which is worth reading. Essentially, we wish to find a region of squared Mahlanobis distance such that:

$$Pr\left((\bar{\boldsymbol{x}} - \boldsymbol{\mu})^T \boldsymbol{S}^{-1} (\bar{\boldsymbol{x}} - \boldsymbol{\mu})\right) c^2)$$

and we can find c^2 as follows:

$$c^2 = \left(\frac{n-1}{n}\right)\left(\frac{p}{n-p}\right) F_{(1-\alpha),p,(n-p)}$$

where $F_{(1-\alpha),p,(n-p)}$ is the $(1-\alpha)$ quantile of the F distribution with p and $n-p$ degrees of freedom, p represents the number of variables and n the sample size.

This illustration is based on, but differs from code provided by Marco Bee to accompany and will make much more sense if used in conjunction with that book. It is worth checking how and why this code differs! Firstly, we need a function to draw ellipses:

```
ellipse <- function(covmat, centroid, csquare, resolution, plot = TRUE) {
angles <- seq(0, by = (2 * pi)/resolution, length = resolution)
  sd <- covmat[1,2] / sqrt(covmat[1,1] * covmat[2,2])
    projmat <- matrix(0,2,2)
    projmat[1,1] <- sqrt(covmat[1,1] %*% (1+sd)/2)
    projmat[1,2] <- -sqrt(covmat[1,1] %*% (1-sd)/2)
    projmat[2,1] <- sqrt(covmat[2,2] %*% (1+sd)/2)
    projmat[2,2] <- sqrt(covmat[2,2] %*% (1-sd)/2)
circle <- cbind(cos(angles), sin(angles))
ellipse <- t(centroid + sqrt(csquare) * projmat %*% t(circle))
if (plot == TRUE) {lines(ellipse)}
return(ellipse)
}
```

It is possible to define a function which calculates c^2 and calls the ellipse routine (I'm not completely convinced this is doing the calculation correctly yet, in particular I'm not sure I'm using the correct tail).

```
function (data, alpha=0.05, resolution=500)
{
xbar <- colMeans(data)
n <- dim(data)[1]
p <- dim(data)[2]
f <- qf(1-alpha, p, n-p)
csquare <- ((n-1)/n) * (p / (n-p)) * f
cat(csquare)
ellipse <- ellipse(cov(data), xbar, csquare, resolution)
}
```

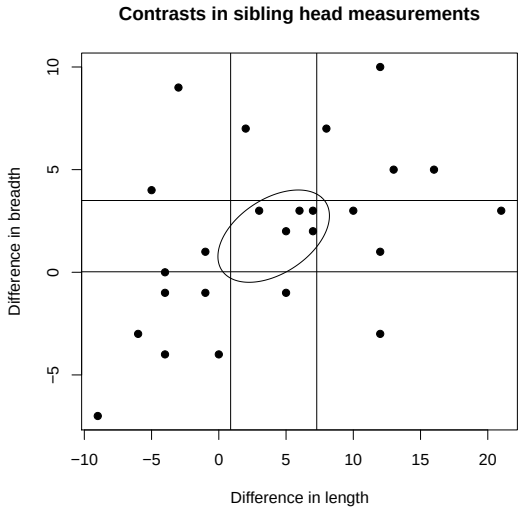


Figure 7.2: Constant density ellipse for mean vector for difference in head width and breadth, and univariate confidence intervals for the mean of each variable

For illustrative purposes, we'll create a $n \times 2$ data object from our flea beetles, and plot the confidence ellipse for these.

```
X <- cbind(flea.beetles[,2], flea.beetles[,3])
plot(X)
cdellipse(X, alpha = 0.01)
cdellipse(X, alpha = 0.05)
```

These can be contrasted with the univariate confidence intervals:

```
abline(v = confint(lm(X[,1]~1)))
abline(h = confint(lm(X[,2]~1)))
```

This exercise should be repeated with the turtles data! However, it is possible to illustrate the basic idea with the sibling heads data, where we construct two derivative variables indicating the difference in head breadth and the difference in head width. These are plotted in figure 7.2, it can be seen that the univariate confidence intervals and the constant density ellipse support different areas of parameter space. Ignoring the correlation structure in these data could lead to flaws in inference when assessesing parameter uncertainty.

7.3 Multivariate Analysis of Variance

As with the univariate situation, t-tests are fine for comparing the means of two groups, but we would have "multiple comparison" problems if we tried to compare more than two. In an analagous way, in the multivariate context we have MANOVA.

First of all we'll enter some data relating to the production of plastic film reported in Krzanowski (2000). Tear, gloss and opacity are measures of the manufactured films.

```
> tear <- c(6.5, 6.2, 5.8, 6.5, 6.5, 6.9, 7.2, 6.9, 6.1, 6.3,
            6.7, 6.6, 7.2, 7.1, 6.8, 7.1, 7.0, 7.2, 7.5, 7.6)
> gloss <- c(9.5, 9.9, 9.6, 9.6, 9.2, 9.1, 10.0, 9.9, 9.5, 9.4,
             9.1, 9.3, 8.3, 8.4, 8.5, 9.2, 8.8, 9.7, 10.1, 9.2)
> opacity <- c(4.4, 6.4, 3.0, 4.1, 0.8, 5.7, 2.0, 3.9, 1.9, 5.7,
               2.8, 4.1, 3.8, 1.6, 3.4, 8.4, 5.2, 6.9, 2.7, 1.9)
Y <- cbind(tear, gloss, opacity)
```

We now need to put in information on the rate of extrusion, and the amount of additive used (`gl()` is a command which specifically creates these kind of experimental factors).

```
> rate <- factor(gl(2,10), labels=c("Low", "High"))
> additive <- factor(gl(2, 5, len=20), labels=c("Low", "High"))
```

There are three conventional ANOVA that could be considered here, but to consider the three responses together we may wish to conduct a MANOVA. However, we can use `manova()` to fit the multivariate ANOVA, and use `summary.aov()` to extract the results of the unvariate analyses.

There are three matrices of interest in MANOVA:

- Total SSP ($\boldsymbol{T}$)
- Between-group SSP ($\boldsymbol{B} = \boldsymbol{T} - \boldsymbol{W}$)
- Within-group SSP ($\boldsymbol{W}$)

Wilk's Lambda is the ratio $\frac{|\boldsymbol{W}|}{|\boldsymbol{T}|}$

```
>      fit <- manova(Y ~ rate * additive)
>      summary.aov(fit)           # univariate ANOVA tables
 Response tear :
              Df  Sum Sq Mean Sq F value   Pr(>F)
rate           1 1.74050 1.74050 15.7868 0.001092 **
```

```
additive       1 0.76050 0.76050  6.8980 0.018330 *
rate:additive  1 0.00050 0.00050  0.0045 0.947143
Residuals     16 1.76400 0.11025
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response gloss :
              Df  Sum Sq Mean Sq F value  Pr(>F)
rate           1 1.30050 1.30050  7.9178 0.01248 *
additive       1 0.61250 0.61250  3.7291 0.07139 .
rate:additive  1 0.54450 0.54450  3.3151 0.08740 .
Residuals     16 2.62800 0.16425
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response opacity :
              Df Sum Sq Mean Sq F value Pr(>F)
rate           1  0.421   0.421  0.1036 0.7517
additive       1  4.901   4.901  1.2077 0.2881
rate:additive  1  3.961   3.961  0.9760 0.3379
Residuals     16 64.924   4.058
```

A call to `summary()` will give the MANOVA table. If you leave out the `text = Ẅilks¨` call you will get a default Pillai-Bartlett statistic.

```
>      summary(fit, test="Wilks") # ANOVA table of Wilks' lambda
              Df  Wilks approx F num Df den Df   Pr(>F)
rate           1 0.3819   7.5543      3     14 0.003034 **
additive       1 0.5230   4.2556      3     14 0.024745 *
rate:additive  1 0.7771   1.3385      3     14 0.301782
Residuals     16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As with Hotellings T^2, Wilk's Lambda has to be converted into an F statistic, a calculation that has been done by the software. It is possible to approximate this by a χ^2_{pbdf} with $W = -\left(w - \frac{p-b+1}{2}\right) log\Lambda$ where p = number of variables, w = residual degrees of freedom (16), b = number of hypotheses degrees of freedom (1).

In any case, the interaction term is not significant. We can fit the model without interactions, which as anticipated suggests that both additive and extrusion rate have an effect on the outcome measures. We will use `by()` to examine the various group means.

```
> fit <- manova(Y ~ rate + additive)
> summary(fit, test = "Wilks")
          Df  Wilks approx F num Df den Df   Pr(>F)
rate       1 0.3868   7.9253      3     15 0.002120 **
```

```
additive   1 0.5538   4.0279      3     15 0.027533 *
Residuals 17
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> by(Y, rate, mean) ## group means according to extrusion rate
INDICES: Low
   tear   gloss opacity
   6.49    9.57    3.79
------------------------------------------------------------
INDICES: High
   tear   gloss opacity
   7.08    9.06    4.08
> by(Y, additive, mean) ## group means according to additive
INDICES: Low
   tear   gloss opacity
   6.59    9.14    3.44
------------------------------------------------------------
INDICES: High
   tear   gloss opacity
   6.98    9.49    4.43
>
> by(Y, list(rate,additive), mean) ## group means by both.
: Low
: Low
   tear   gloss opacity
   6.30    9.56    3.74
------------------------------------------------------------
: High
: Low
   tear   gloss opacity
   6.88    8.72    3.14
------------------------------------------------------------
: Low
: High
   tear   gloss opacity
   6.68    9.58    3.84
------------------------------------------------------------
: High
: High
   tear   gloss opacity
   7.28    9.40    5.02
```

High levels of extrusion rate lead to higher levels of tear and opacity but lower levels of gloss. High levels of addtive lead to higher levels of tear, gloss and opacity.

Chapter 8

Discriminant analysis

Discriminant analysis presupposes that we have a number of known groups of individuals, and a set of data which has been collected on individuals within these groups. We wish to find a way of using that data to predict which group these individuals belong to, either to understand the differences or to be able to predict group membership. Bumpus (1898) collected data on sparrows who survived and didn't survive a storm, these data are extensively analysed in this context by Manly, the primary aim of the analysis being to look for differences between the groups. Are there different features which help us tell storm survivors from non-survivors? More usually, we may be interested in predicting group membership. Common examples can be found in finance; can banks tell good credit risks from bad based on data collected on customers who have subsequently defaulted on loans, see Johnson and Wichern (1998) for more details. Another good account of discriminant analysis is given by Flury (1997) who suuggests it may be valuable when we have to carry out destructive procedures to determine group membership (such as in certain quality control investigations). Finally a rather brief account is given in Venables and B.D.Ripley (2002), which gives the example of disease diagnosis. Consider a set of measurements of patient characterstics, and information determined on whether these patients have breast cancer or not. We would be very interested in being able to make a determination of breast cancer based on the data, rather than having to wait for biopsy or other pathological information.

Discriminant analysis in one dimension seems straightforward enough. We can examine the densities of the two groups and find an optimal cut-off point, which classifies the two groups as accurately as possible. Some idea of the procedure is given in figure 8.1, which illustrates the idea behind discriminant function.

Note immediately that there is a measureable risk of misclassification, which depends on the variance within groups and the separation between groups. All we need to do is extent this procedure to work in more than one dimension. We are going to realise this by seeking a linear combination giving us

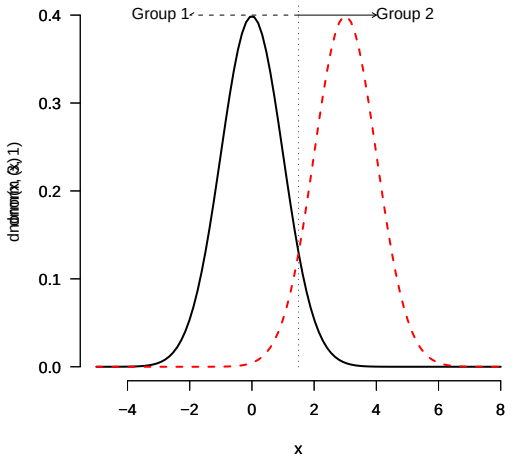


Figure 8.1: Idealised discrimant function

the largest separation between groups. In other words, we are going to find linear combinations based on the original variables:

$$z = a_1x_1 + a_2x_2 + \ldots + a_px_p \tag{8.1}$$

However our linear combination this time will be optimised to give us the greatest potential for distinguishing the two groups. Having found a suitable linear combination, we select a cut-off point (denoted by the vertical dotted line in the figure above), and assign observations to group 1 or group 2 based on the value relative to the cut-off. You can see from the stylised function shown that some observations will be misclassfied! We check the performance of this aspect of our procedure by means of a confusion matrix.

Recall that when conducting the T^2 test we essentially looked for linear combination of variables which maximised the difference between groups. Similar ideas apply in discriminant analysis. We seek a transformation of the data which gives the maximum ratio of group means to group variance within the two groups, i.e. we are maximising the between group variation relative to the within group variance - this should sound vaguely like what goes on in ANOVA:

Source	d.f.	Mean Square	F ratio
Between groups	$m-1$	M_B	M_B/M_W
Within groups	$N-m$	M_W	
	N-1		

We need to find a linear combination that yields as large an F ratio as possible, hence the coefficients

$a_1, \ldots, a_p$ need to be chosen to maximise this value. More than one discriminant function is available, there are

$$s = min(p, m-1) \tag{8.2}$$

discriminant functions available, where p is the number of variables, and m is the number of groups.

Considering the case where we have $m > 2$ groups, and $p > 2$ variables, we are looking for the following discriminant functions:

$$\begin{aligned} z_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p \\ z_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2p}x_p \\ \ldots & \\ z_s &= a_{s1}x_1 + a_{s2}x_2 + \ldots + a_{sp}x_p \end{aligned}$$

although hopefully only a small number of these linear combinations will account for all important differences between groups.

8.1 Fisher discimination

Remember the T^2 statistic:

$$T^2(\boldsymbol{a}) = \frac{\left(\boldsymbol{a}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)\right)^2 n_1 n_2/(n_1+n_2)}{\boldsymbol{a}^T \boldsymbol{S} \boldsymbol{a}} \tag{8.3}$$

This is equivalent to finding $\boldsymbol{a}$ which maximises $|\boldsymbol{a}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)|$ subject to $\boldsymbol{a}^T \boldsymbol{S} \boldsymbol{a} = 1$. This has a single solution:

$$\boldsymbol{a} = \boldsymbol{S}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$$

and so the linear discriminant function is given by:

$$z = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T \boldsymbol{S}^{-1} \boldsymbol{x}$$

In two dimensions, an obvious cut-off would be the midpoint between the mean value of z for group 1 and 2.

Fisher's approach has been extended to cope with more than two groups. Again, we wish to find a linear combination $z = \boldsymbol{a}^T\boldsymbol{x}$ which maximised the ratio of between group variance to within group variance.

If we calculate the within sample matrix of sum of squares and cross products $\boldsymbol{W}$, and the total sample matrix of sum of squares and cross products $\boldsymbol{T}$, we can easily find the between-groups sample matrix sum of squares and cross products:

$$\boldsymbol{B} = \boldsymbol{T} - \boldsymbol{W} \tag{8.4}$$

In effect we wish to maximise:

$$\frac{\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}}{\boldsymbol{a}^T\boldsymbol{W}\boldsymbol{a}} \tag{8.5}$$

and usually do this subject to the condition that $\boldsymbol{a}^T\boldsymbol{W}\boldsymbol{a} = 1$. Fisher's method of discriminant analysis reduces to finding the eigenvalues and corresponding eigenvectors of $\boldsymbol{W}^{-1}\boldsymbol{B}$. The ordered eigenvalues $\lambda_1, \ldots, \lambda_s$ are the ratio of between groups to within groups sum of squares and cross products for $z_1, \ldots, z_s$, the corresponding eigenvectors, $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_s$, where $\boldsymbol{a}_i = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{ip} \end{pmatrix}$ are the coefficients of z_i.

We make a number of big assumptions in discriminant analysis: that observations are a random sample, that they are normally distributed and that the variance is the same for each group. Discriminant analysis is relatively resistant to some departures from the normality assumption - it can cope with skewness but not with outliers.

8.2 Accuracy of discrimination

Clearly, one important measure of the success of our discriminant rule is the accuracy of group prediction: note that there are a number of ways of measuring this and that discriminant analysis is one technique among many used in *supervised classification*. There are many techniques in machine learning and other areas which are used for classification, for example you have already met the technique of logistic discrimination.

Model over-fitting is a known problem: we can fit a classifier really really well to our existing data but

it doesn't work well next time we carry out a data collection exercise. A key concept in this regard is the use of training and testing sets, where we split our data into two groups, and use one part to build a classifier, and the other to test it. There are many other technques which can help in this regard, for example leave one out (loo) cross validation and some of the more recent multivariate texts should be consulted.

An important idea in terms of measuring the success of our classifier is the *confusion matrix*. This sounds rather grand, but is basically a matrix telling us how many times our discriminant function made a correct classification, and how many times it got it wrong.

8.3 Importance of variables in discrimination

Some textbooks refer to questions surrounding selection of variables for use in a classifier. It is important to consider whether variables are necessary for classification; often a tolerance test may be used prior to the analysis to remove multicollinear and singular variables. It may even be desirable to carry out a dimension reducing technique. The reason for carrying out these tests are related to over-fitting.

Some software provides "standardised" coefficients, the idea being that perhaps it is safe to remove variables with small standardised coefficients. However, another approach could well be to consider classifiers with different numbers and combinations of variables and contrast the confusion matrix. This might help identify variables which do the best job of distinguishing the groups, and those which are the least necessary.

8.4 Canonical discriminant functions

As mentioned earlier, we can have more than one discriminant function. It is usual to plot these on a scatterplot in an attempt to visualise the discriminant ability. However, there are tests of significance.

For example, we wish to find discriminants with a small Wilk's lambda ($\frac{|\boldsymbol{W}|}{|\boldsymbol{T}|}$), in our case this can be derived as :

$$\Lambda^2 = \left(\sum_{k=1}^{m} n_k - 1 - \frac{1}{2}(p+m) \right) \ln(1+\lambda_j), \tag{8.6}$$

which has a χ^2 distribution with $p + m - 2j$ degrees of freedom.

8.5 Linear discrimination - a worked example

In practice, we're going to consider a classification exercise on the Iris data. This is rather well known data featuring three species of Iris, and four anatomical measures. First of all we need to load the `MASS` to obtain the `lda()` function. Then we are going to pull out a training set from the Iris data.

```
> library(MASS)
>  data(iris3)
>  Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
+                          Sp = rep(c("s","c","v"), rep(50,3)))
>      train <- sample(1:150, 75)
```

`train` is a set of index numbers which will allow us to extract a training set. We use `lda()` to fit a discriminant analysis, setting all priors equal to 1 (i.e. group memberships the same), and `subset = train` to fit the analysis to the training set. The squiggle dot indicates that we wish to use all other variables within Iris to predict Species (Sp).

```
> z <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)
> z
```

Having extracted a training set, we are going to classify the remaining Iris' and see how well the predicted and actual species line up.

```
> actual <-  Iris[-train,]$Sp
> preds <- predict(z, Iris[-train, ])$class)
> xtabs(~actual + preds)
```

One little thing to watch when using software is that in practice, Fisher's approach tends not to be used. An approach based on probability distributions and using Bayes rule is common. All this does, is correct for the proportions in each group to start with. Instead of finding a discriminant rule assuming a 50:50 split, we use information on more plausible group numbers.

Chapter 9

Principal component analysis

The origins of principal component analysis are usually traced to Pearson (1901) who was concerned with the fitting planes in the analysis of covariance matrices by orthogonal least squares. Much development of the technique is ascribed to Hotelling (1933), who, working with the correlation matrix provided another development of the technique which is more familiar. As a technique in its own right it has received book length treatment by Jackson (1991) and Jolliffe (1986); another excellent exposition is given in Flury (1988) who also develops one generalisation of the technique to multiple groups. The theory underlying principal components is important in a wide range of areas, consequently this chapter will examine some of the underlying theory as well as considering conventional approaches to principal component analysis. An **R**-centric selection of recent extensions to the technique will also be considered in the next chapter.

Principal component analysis can be performed for a variety of reasons, one can perhaps consider its use in one of three ways. Arguably the most common use is in terms of dimension reduction. Principal component analysis can be thought of as a data analytic method which provides a specific set of projections which represent a given data set in fewer dimensions. This has obvious advantages when it is possible to reduce dimensionality to two or three as visualisation becomes very straightforward but it should be acknowledged that reduced dimensionality has advantages beyond visualisation. Another rationale for conducting principal components analysis is to transform correlated variables into uncorrelated ones, in other words to *sphere* the data. Whilst univariate data can be standardised by centering and scaling, in a multivariate context one may also wish to "standardise" the correlation between variables to zero. The final rationale for the technique is that it finds linear combinations of data which have relatively large (or relatively small) variability.

In essence, we wish to form projections $\boldsymbol{z} = (z_1, z_2, \ldots, z_q)$ of the *standardised* data $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$

$$
\begin{aligned}
z_1 &= e_{11}x_1 + e_{12}x_2 + \ldots e_{1p}x_p \\
z_2 &= e_{21}x_1 + e_{22}x_2 + \ldots e_{2p}x_p \\
\ldots & \\
z_q &= e_{q1}x_1 + e_{q2}x_2 + \ldots e_{qp}x_p
\end{aligned}
$$

where the coefficients $\boldsymbol{E}$ form a $p \times q$ matrix. If dimension reduction is our goal, clearly we hope that we can form an adequate representation of the data with $q < p$.

We are going to illustrate Principal Component Analysis by reference to three sets of data. The first was presented by Karlis et al. (2003) and relates to Heptathalon results from the Sydney Olympics in 2000. In this event, athletes compete in seven different sporting activities, and clearly for the purposes of the competition a decision has to be made as to who is the best heptathelete overall. For sporting purposes, points are awarded for performance in each event and summed. In essence, a one dimensional composite variable has to be created from the results for the seven events before a decision can be made as to who wins the medals. We are not necessarily going to challenge the way the International Olympic Committee calculates scores, but clearly a linear combination which achieved maximum separation of athletes might be interesting. A trivial linear combination would be to take $\boldsymbol{e} = \frac{1}{n}\mathbf{1}$. However, the aim of any projection method is to find an *interesting* projection of the data. All we need to do is decide what we mean by *interesting*. As mentioned earlier, one use of principal components is to find projections where the variance is maximised, thus maximising the separation between heptatheletes.

The second rather well used set of data relates to carapace measurements the painted turtle *Chrysemys picta marginata* first reported by Jolicoeur and Mosimann (1960). These data contain 3 variables recording the shell length, width and height for 24 male and 24 female turtles. We will consider these data partly because the turtles are cute, but mainly because it affords some consideration of the role of standardisation. Standardising by subtracting mean and dividing by the standard deviation can be a rather brutal approach, it is possible to take the natural logarithms of these anatomical measures which allows a different insight into the relationship between the measures.

Finally, we will investigate some gene expression data. In microarray experiments, one typically collects data from a relatively small number of individuals, but records information on gene expression of a large number of genes and therefore have a data matrix $\boldsymbol{X}$ where $n << p$. Dimension reduction is a central concern at some stage of the analysis, either on the whole set of genes or on a selected subset. As might become clear, groups of genes which can be identified analytically as co-expressing are referred to as eigengenes. We will therefore consider some of the issues surrounding the interpretation of principal components in high dimensions.

Some other data sets will be included to illustrate specific points where necessary, and the excercises

will also develop other data sets.

9.1 Derivation of Principal Components

We will firstly consider the derivation of population principal components in the style proposed by Hotelling (1933) as a way of finding linear combinations with maximum variance. Given that we have a random vector $\boldsymbol{x} \in \mathbb{R}^P$, we wish to consider the vector of transformed variables $\boldsymbol{z} = \boldsymbol{\alpha}^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$, and so we wish to find α such that the Var(z) is maximised. By convention, we insist that $\boldsymbol{\alpha}$ is orthonormal, i.e. in addition to being a vector of unit length we require that $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$. To find an expression for Var(z) in terms of the original variables $\boldsymbol{x}$, firstly by considering the following relationship:

$$\text{Var}(\boldsymbol{z}) = \text{Var}(\boldsymbol{\alpha}^T\boldsymbol{x}) = E(\boldsymbol{\alpha}^T\boldsymbol{x})^2$$

Hence,

$$\sum_{i+1}^{n}(\boldsymbol{z}_i - \bar{\boldsymbol{z}})^2 = \sum_{i=1}^{n}\boldsymbol{\alpha}^T\left[(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\right]\boldsymbol{\alpha}$$

and premultiplying both sides by $\frac{1}{n-1}$ gives us an estimate of the variance of the transformed variables in terms of the original variables. To find $\boldsymbol{\alpha}$ in order to maximise the variance we therefore wish to find:

$$\max(\boldsymbol{\alpha}^T\hat{\boldsymbol{\Sigma}}\boldsymbol{\alpha}) \tag{9.1}$$

subject to the side condition that $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$. Considering the first principal component, we wish to maximise Var$(z_1) = \boldsymbol{\alpha}_1^T\hat{\boldsymbol{\Sigma}}\boldsymbol{\alpha}_1$; it can be shown that this problem can be specified by incorporating a Lagrange multiplier. Consider maximising the expression φ given below, where λ is a Lagrange multiplier:

$$\varphi_1 = \boldsymbol{\alpha}_1^T\hat{\boldsymbol{\Sigma}}\boldsymbol{\alpha}_1 - \lambda_1(\boldsymbol{\alpha}_1^T\boldsymbol{\alpha}_1 - 1) \tag{9.2}$$

which can be solved by differentiation with respect to $\boldsymbol{\alpha}_1$. The differential gives us a system of linear equations as follows:

$$\frac{\partial\varphi_1}{\partial\boldsymbol{\alpha}_1} = 2\Sigma\boldsymbol{\alpha}_1 - 2\lambda\boldsymbol{\alpha}_1 \tag{9.3}$$

and setting the derivative equal to zero gives:

$$\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_1 \tag{9.4}$$

which is easily rearranged to give:

$$(\boldsymbol{\Sigma} - \lambda_1\boldsymbol{I})\boldsymbol{\alpha} = 0. \tag{9.5}$$

We have p equations and p unknowns, but we can find non-trivial solutions where $\boldsymbol{\alpha} \neq \boldsymbol{0}$ noting that

if $\boldsymbol{a}_1^T\boldsymbol{a} = 1$ then $\boldsymbol{\Sigma} - \lambda\boldsymbol{I}$ is singular which therefore implies that:

$$|\boldsymbol{\Sigma} - \lambda_1\boldsymbol{I}| = 0 \tag{9.6}$$

This looks like an eigenproblem, so λ_1 can be found as the largest eigenvalue of $\boldsymbol{\Sigma}$, and $\boldsymbol{\alpha}_1$ as the corresponding eigenvector.

For the second principal component, as we are going to assume orthogonality, we wish to add the side condition that $\text{Cor}(z_1, z_2) = 0$. We therefore need to solve:

$$\varphi_2 = \boldsymbol{\alpha}_2^T\hat{\boldsymbol{\Sigma}}\boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2^T\boldsymbol{\alpha}_2 - 1) - \mu(\alpha_1^T\boldsymbol{\Sigma}\boldsymbol{\alpha_2}) \tag{9.7}$$

where differentiating now gives:

$$\frac{\partial\varphi_2}{\partial\boldsymbol{\alpha}_2} = 2\Sigma\boldsymbol{\alpha}_2 - 2\lambda_2\boldsymbol{\alpha}_2 - \mu\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 \tag{9.8}$$

but as we assume $\text{Cor}(z_1, z_2) = 0$ we take $\boldsymbol{\alpha}_2^T\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_1^T\boldsymbol{\alpha}_2 = 0$ implying that $\boldsymbol{\alpha}_1^T\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2^T\boldsymbol{\alpha}_1 = 0$ hence we can premultiply our equation in 9.8 by $\boldsymbol{\alpha}_2$ giving:

$$2\boldsymbol{\alpha}_2^T\Sigma\boldsymbol{\alpha}_2 - 2\boldsymbol{\alpha}_2^T\lambda_2\boldsymbol{\alpha}_2 - \mu\boldsymbol{\alpha}_2^T\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 \tag{9.9}$$

which implies that $\mu = 0$, and that λ_2 is also an eigenvalue of $\boldsymbol{\Sigma}$.

Clearly in any formal sense we should continue this process, but the apparent pattern can be demonstrated. To find further principal components we need to take the spectral decomposition of the covariance matrix, subject to normalising the eigenvectors to have unit length. In order to find principal components which have maximum variance subject to the condition of being orthogonal to any previous principal component we only need to solve:

$$|\boldsymbol{\Sigma} - \lambda\boldsymbol{I}| = 0$$

and to find:

$$\boldsymbol{\Sigma}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$$

Of some interest to us is that if any matrix $\boldsymbol{\Sigma}$ is symmetric (which will always be the case for correlation and covariance matrices), the normalised eigenvectors corresponding to unequal eigenvalues are orthonormal.

In practice, we don't know Σ and we have to estimate it from our data with a suitable estimate. We can consider a number of possibilities, indeed we will examine robust estimators later. However, most development of principal components assume the unbiased estimator $\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X}$, where $\boldsymbol{X}$ is a matrix of mean centred data, although we will note later that one of the **R** functions actually uses

$\boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}$. $\boldsymbol{\mu}$ is readily estimated from the sample mean. However, before considering applications to sample principal components, we make a few comments on the geometry of this technique.

9.1.1 A little geometry

Whilst we have covered the variance maximising property of principal components, it is possible to consider the technique from a geometric perspective. Geometry is perhaps rather more important in recent developments in principal components, details on the concept of *self-consistency* are given by Flury (1997). For now, we note that from a geometrical perspective we wish to minimise the perpendicular distance between points and the new projection, the same problem Pearson (1901) was solving.

Consider the vector of observations $\boldsymbol{x} = (x_1, \ldots, x_p) \in \mathbb{R}_p$ We want to project these obsevations onto $\lambda\boldsymbol{\alpha}$, and in doing so to find the best fitting q dimensional subspace. Denoting the projection of $\boldsymbol{x}$ by $P\boldsymbol{x}$, we therefore wish to minimise:

$$(\boldsymbol{z} - P\boldsymbol{x})^T(\boldsymbol{z} - P\boldsymbol{x})$$

Ideally, we wish the projection to be performed in terms of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_q)$, although with a p dimensional representation this can clearly take the form $(\alpha_1, \ldots, \alpha_q, \alpha_{q+1}, \alpha_p)$.

The projection can be given as

$$P\boldsymbol{x} = \boldsymbol{x}\boldsymbol{\alpha}^T\boldsymbol{\alpha}$$

where $\boldsymbol{\alpha}^T\boldsymbol{\alpha}$ can be described as the projection matrix. We can rewrite this as:

$$P\boldsymbol{x} = \sum_1^q (x_i\boldsymbol{\alpha}_i^T)\boldsymbol{\alpha}_i.$$

We can therefore rewrite $\boldsymbol{z} - P\boldsymbol{x}$ as

$$\boldsymbol{z} - P\boldsymbol{x} = \sum_{q+1}^p (x_i^T\boldsymbol{\alpha}_i)\boldsymbol{\alpha}_i$$

Clearly

$$(\boldsymbol{z} - P\boldsymbol{x})^T(\boldsymbol{z} - P\boldsymbol{x}) = \sum_{q+1}^p (x_i^T\boldsymbol{\alpha}_i)^2$$

Consequently, we wish to minimise

$$\sum_{j=1}^{p}\sum_{i=1}^{n}(x_j^T\alpha_i)^2$$

Which is Pearson's orthogonal least squares problem. However, we can find a link with Hotelling's approach by noting that $\sum_{j=1}^{n} x_j^T x_j = \sum_{j=1}^{p}\sum_{i=1}^{n}(x_j^T a_i)^2$ our problem can also be expressed as one where we wish to maximise

$$\sum_{j=1}^{p}\sum_{i=1}^{n}(x_j^T\boldsymbol{\alpha}_i)^2$$

Noting that this last term is $\sum_{i=1}^{q}\boldsymbol{\alpha}_i^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\alpha}_i$, it might be apparent that we actually seek to find $\boldsymbol{\alpha}$, subject to $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = \delta$ to maximise $\boldsymbol{\alpha}^T\boldsymbol{\Sigma}\boldsymbol{\alpha}$ which looks like a familiar problem!

It may now be noted that the eigenvectors are related to the angle between the untransformed data and the principal component. Assuming we have $j = 1, \ldots, p$ variables, and $k = 1, \ldots, p$ projections, this angle is given by:

$$\cos\theta_{jk} = \alpha_{jk} \tag{9.10}$$

This is reasonably convenient to plot in 2-dimensions. Given simulating bivariate data $\boldsymbol{x} = (x_1, x_2)$, the angle between x_1 and the first principal component is given by $\cos\theta_{11} = \alpha_{11}$. Rather conveniently, this is implicit in the slope `b=` supplied to `abline()`. An illustration of orthogonal projection can be explored using the following code. This should be contrasted with the least squares fit.

```
require(MASS)
X <- scale(mvrnorm(25, c(2,2), matrix(c(1,0.8,0.8,1),2,2)))
eqscplot(X)
X.cov <- cov(X)
X.ed <- eigen(X.cov)
proj <- X.ed$vec[,1] %*% t(X.ed$vec[,1])
y <- t(proj %*% t(X))
abline(a=0,b=X.ed$vec[2,1]/X.ed$vec[1,1])
arrows( X[,1], X[,2], y[,1],y[,2], length = 0.05, col = "green")
```

Plotting the ordinary least squares solutions is easy enough, although when regression X[,1] on X[,2] the gradient needs to be inverted (or the axes reversed).

```
## plot the ols of X[,2] on X[,1]
eqscplot(X)
X2.lm <- lm(X[,2] ~ X[,1])
abline(X2.lm, col = "red", lwd = 2)
arrows(X[,1],X[,2], X[,1], predict(X2.lm),  length = 0.01, col = "red")
points(X[,1],X[,2], pch = 16)
```

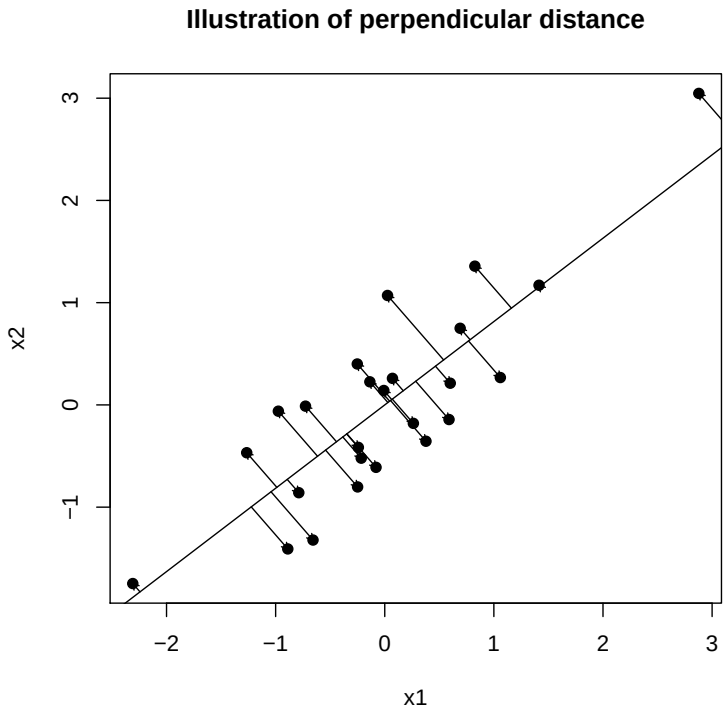


Figure 9.1: Illustration of perpendicular distance

```
## plot the ols of X[,1] on X[,2]
eqscplot(X)
X1.lm <- lm(X[,1] ~ X[,2])
abline(0, 1/coef(X1.lm)[2], col = "blue")## need to invert the gradient
arrows(X[,1],X[,2], predict(X1.lm), X[,2],  length = 0.01, col = "blue")
points(X[,1],X[,2], pch = 16)
```

It is informative to contrast the line plotted in figure 9.1 with those produced by linear regression of x on y as well as y on x. The matrix $\boldsymbol{a}\boldsymbol{a}^T$ can be referred to as the projection matrix.

9.1.2 Principal Component Stability

Whilst considering the geometry, it is useful to motivate some ideas about the differences between population and sample principal components by simulating three datasets from the same parameters. For population 1, we draw from standard normal variables with correlation of 0.9, for population 2 the variables are uncorrelated. In other words, $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = (0,0)^T$, $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$ but $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Simulating the data is simple enough, for example:

```
X <- mvrnorm(100, c(0,0), matrix(c(1,.9,.9,1),2,2))
V <- var(X)
eV <- eigen(V)
```

```
plot(X, xlim = c(-3,3), ylim = c(-3,3))
abline(a=0,b=eV$vec[2,1]/eV$vec[1,1])
abline(a=0,b=eV$vec[2,2]/eV$vec[1,2])
```

Just so the eigenvalues don't feel left out, we can add constant density ellipses to these plots, details on these were given in 7.2. Essentially, we can define a constant density ellipse from the "centre" as $\pm c\sqrt{\lambda_i}\boldsymbol{\alpha}_i$, here we take $c = 1.96$. In the code snippet below, x and y are the coordinates of an ellipse scaled from a unit circle by the two eigenvalues. These are then rotated by the angles of the eigenvectors:

```
theta <- seq(0,(2*pi),length=101)
x <- 1.96 * sqrt(eV$val[1]) * cos(theta)
y <- 1.96 * sqrt(eV$val[2]) * sin(theta)
newxy <- cbind(x,y) %*% t(eV$vec)
lines(newxy)
```

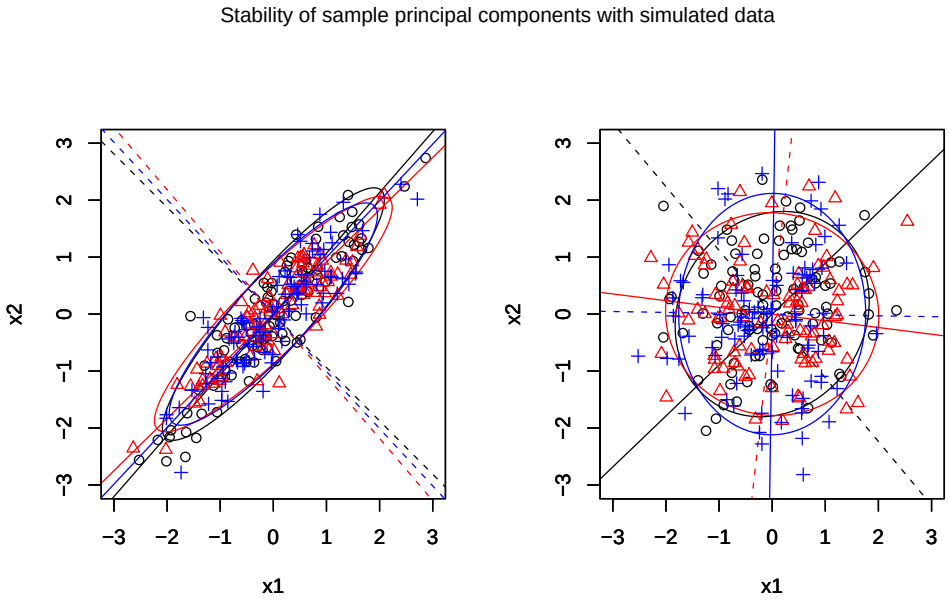


Figure 9.2: Artificially generated data indicating stablity of pca solution

We now consider figure 9.2, which overlays three samples from each of the two populations specified. The semi-major axis, the contribution to the first principal component has been denoted with solid lines, the semi-minor axis, the contribution to the second principal component, has been denoted by dotted lines. It should be very apparent that there is some variation in the orientation of the axes, as might be expected the eigenanalysis varies slightly according to sample properties. However, the right hand side of the plot is intended to illustrate the problem of sphericity. Without a strong correlation structure, the angles subtended by the principal components varies massively.

Whilst this is an artificial example in many senses (not only are the data simulated but we have only two variables), but the problem is an important one. In particular, when exploring dimension reduction properties of principal components we have to be attendant to the possibility of partial sphericity; that some of the $q+1,\ldots,p$ principal components essentially exhibit this behaviour. We consider specific hypothesis tests for this problem later.

Finally, whilst talking about projections, we make one comment in relation to projection pursuit. This technique will be considered further in the next chapter. Suffice to say here that projection pursuit is a generic set of techniques which aims to find "interesting" projections of data. The user decides on the dimensionality of the projection and selects a suitable criterion of interestingness. Bolton and Krzanowski (1999) give results interpreting principal components analysis within this framework, the criterion (projection pursuit index) in this case is given by:

$$I = max(\boldsymbol{e}\boldsymbol{S}\boldsymbol{e}^T); \boldsymbol{e}\boldsymbol{e}^T = 1,$$

where $\boldsymbol{S}$ is the sample covariance matrix. This index is the minimum of the maximised log-likelihood over all projections when normality is assumed.

However, they go on to imply that more "interesting" projections are less likely to have normally distributed data by showing that in terms of the likelihood $\mathscr{L}(\boldsymbol{e})$, this decreases as $\boldsymbol{eSe}^T$ increases. Assuming the usual maximum likelihood estimators for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$\begin{aligned}\mathscr{L}(\boldsymbol{e}) &= max\mathscr{L}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{e}) \\ &= -\frac{n}{2}\left[p + p\log\left(\frac{2\pi n}{n-1}\right)\right] - \frac{n}{2}\log|\boldsymbol{eSe}^T|\end{aligned}$$

In other words, under normality, the most "interesting" projection is the one with the maximised likelihood.

9.2 Some properties of principal components

Before considering some examples of principal component analysis we first consider a number of fundamental properties.

Definition 9.3 *If $\boldsymbol{x}$ is a mean centred random vector, with covariance matrix $\boldsymbol{\Sigma}$, the principal components are given by:*

$$\boldsymbol{x} \rightarrow \boldsymbol{z} = \boldsymbol{E}^T(\boldsymbol{x} - \boldsymbol{\mu}) \tag{9.11}$$

i.e. the transformation requires mean-centred variables. $\boldsymbol{E}$ is orthogonal, $\boldsymbol{E}^T\boldsymbol{\Sigma}\boldsymbol{E}$ is diagonal, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$ provided $\boldsymbol{\Sigma}$ is positive definite.

A number of key propeties immediately follow from their derivation from the spectral decomposition:

Theorem 9.4

$$E(z_i) = 0 \tag{9.12}$$

$$Var(z_i) = \lambda_i \tag{9.13}$$

hence:

$$Var(z_1) \geq Var(z_2) \geq \ldots \geq Var(z_p); \tag{9.14}$$

in particular it should be noted that no standardised linear combination of $\boldsymbol{x}$ has a larger variance than $\boldsymbol{\lambda}$.

We finally restate the implications of finding orthogonal projections.

$$Cov(z_i, z_j) = 0, i \neq j; \tag{9.15}$$

One property we will make repeated use of concerns the proportion of variance explained by each principal component:

Theorem 9.5 *The trace of a covariance matrix is equal to the sum of its eigenvalues:*

$$trace(\mathbf{\Sigma}) = \sum_{i=1}^{p} \lambda_i \tag{9.16}$$

In other words, equation 9.16 indicates that the *total variance* can be explained by the sum of the eigenvalues. In the case of principal components formed from the correlation matrix $\boldsymbol{R}$ this will be equal to the number of variables.

Theorem 9.6 *The generalised variance (the determinant of a covariance matrix) can be expressed as the product of its eigenvalues:*

$$|\mathbf{\Sigma}| = \prod_{i=1}^{p} \lambda_i \tag{9.17}$$

Theorem 9.6 indicates that we can find an estimate of the *generalised variance* from the product of the eigenvalues. Along with theorem 9.5 we find that the generalised variance and the sum of variances are unchanged by the principal component transformation.

Finally, a note is needed on scale-invariance. Flury (1997) describes this last as an anti-property. Principal components are not-invariant to changes of scale. Standardising variables is rather a brutal way of dealing with this. Explanations are given in most multivariate textbooks, for example both formal and information explanaitions are given by (Mardia et al., 1979, page 219). Nevertheless, if variables are recorded on widely differing scales, a principal component analysis of the covariance matrix will largely reflect the variables with the numerically greatest variance. It is therefore important that the variables are in some sense comparable; this can either be achieved by standardising, or by some gentler transformation. We will find that the heptathalon data has to be standardised, whereas it is possible to take logs of the turtle data.

Finally, we consider one important property, which is more in the Pearson (1901) sense.

Theorem 9.7 *The first k principal components have smaller mean squared departure from the population (or sample) variables than any other k-dimensional subspace.*

Proof: (Mardia et al., 1979, page 220)

This property is rather important when considering the dimension reducing properties of principal component as it does not require any distributional assumptions.

9.8 Illustration of Principal Components

We define the sample principal components by $\boldsymbol{e}$ and ℓ.

Having now hopefully explained the rationale behind principal components analysis, we consider some illustrative analysis before considering further inferential developments.

9.8.1 An illustration with the Sydney Heptatholon data

Before doing anything else with these data, it needs to be noted that in the three running events, better performance is indicated by a lower measure (time), whereas in the jumping and throwing events good performance is indicated by a higher measure (distance). It seems sensible to introduce a scale reversal so that good performance is in some way at the top of any given scale. A convenient way of doing this is to multiply the times of the running events by -1.

```
hept.df <- read.csv("Heptatholon.csv", row.names = 1)
hept.df$X100mHurdles.S. <- hept.df$X100mHurdles.S. * -1
hept.df$X200m.sec. <- hept.df$X200m.sec. * -1
hept.df$r800m.s. <- hept.df$r800m.s. * -1
```

These variables are clearly incomparably in any sense. It is also clear that we need to work with the correlation matrix for these data, there is considerable difference in the scales (running 800 metres tends to take rather longer than running 100 metres). We will also centre the variables using `scale()` which saves us a little work later on.

```
hep.scale <- scale(hept.df[,-1])
hept.cormat <- cor(hept.df[,-1])
hep.ev <- eigen(hept.cormat)
```

Our principal component analysis then basically consists of extracting `hep.ev$values` contains the eigenvalues, and `hep.ev$vectors` contains the eigen vectors. Our first set of loadings are given by the first row of the eigenvectors, we can form the first linear combination:

```
> hep.ev$vectors[,1]
> z1 <- hep.scale %*% hep.ev$vectors[,1]
```

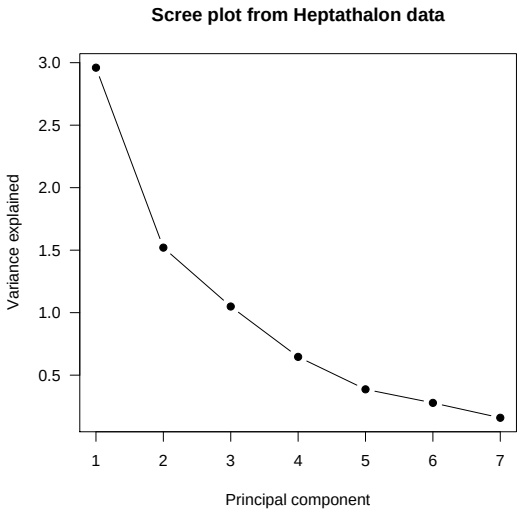


Figure 9.3: Scree plot displaying the amount of variation explained by each of the seven principal components formed from the correlation matrix of the Sydney Heptathalon data

in a similar manner it is possible to form z_2 and so on.

This means that the proportion of total variance explained by each linear combination can be given by $\frac{\lambda_i}{\sum_{i=1}^{p} \lambda_i}$, which can be calculated for our heptathalon data with `hep.ev$values/sum(hep.ev$values)`.
It is also conventional to produce a "scree" plot of this information with something like `plot(hep.ev$values, type = "b")` which graphically represents the amount of variance explained by each linear combination.

9.8.2 Principal component scoring

As stated at the start of the chapter, the principal component scores are essentially derived from the mean centered data.

In other words, we found:

$$\boldsymbol{z} = \boldsymbol{E}^T(\boldsymbol{x} - \boldsymbol{\mu}) \tag{9.18}$$

It is possible (although somewhat unusual in this context) to find scores from the standardised data

$$\boldsymbol{z} = \boldsymbol{E}^T(\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Lambda}^{-1/2} \tag{9.19}$$

It should be noted that Rencher (2002) really doesn't approve of this latter effort, it essentially leaves us looking at the correlation between the standardised data and the principal component scores. We therefore consisder the more usual scores given in equation 9.18. The first four eigenvectors are given as follows:

```
$vectors
           [,1]        [,2]        [,3]         [,4]
[1,] -0.4656494  0.28868133  0.32883275 -0.003922038
[2,] -0.2455859 -0.56442826 -0.10737271 -0.610427608
[3,] -0.4195748 -0.07137064 -0.52173345  0.235891780
[4,] -0.4330174 -0.02204083  0.51825157  0.357022268
[5,] -0.4630436  0.11516723  0.12459693 -0.480637087
[6,] -0.3228125  0.38161226 -0.56832241  0.091073036
[7,] -0.2017272 -0.65849286 -0.03216966  0.452710298
```

So to find the first principal component we only need to compute the following:

$$z_{i1} = a_{11}x_{i1} + a_{12}x_{i2} + a_{13}x_{i3} + \ldots$$

which means that the first principal component can be given as:

$$Z_{i1} = -0.4656 \times x_1 - 0.2456 \times x_2 - 0.4196 \times x_3 + \ldots$$

Clearly this can also be calculated as matrix product $\boldsymbol{Xa}$.

```
hep.ev$vectors[,1] ## first vector of pc loadings
scores <- hep.scale %*% hep.ev$vectors
par(mfrow = c(3,3))
apply(scores, 2, qqnorm)
scoresR <- hep.scale %*% (hep.ev$vectors %*% diag(hep.ev$values^-0.5))
```

If we wished, we can carry out further investigation of the principal component scores.

9.8.3 Prepackaged PCA function 1: `princomp()`

In practice it will come as no surprise to learn that there are prepackaged functions in **R** for carrying out a principal components analysis. `princomp()` has been provided for comparability with S-Plus. We will find out later that the preferred function uses the singular value decomposition. However, there are good reasons for examining `princomp()` first. It is based on carrying out an eigen decomposition,

by default of the covariance matrix and it should be noted that the covariance matrix esimated by `cov.wt` uses the divisor N, rather than the unbiased version $N-1$. It is possible to supply robust estimates of the covariance matrix via `covmat`, this does allow a form of robust principal components and as we work through the heptathalon data we will find out that this may indeed be useful. It is possible to call `princomp()` with the specification `cor=TRUE` to use the sample correlation matrix rather than sample covariance matrix.

The main results can be accessed via `summary()` and `print()`. The eigenvectors are extracted and printed with a degree of pretty printing using `loadings()`. If necessary, the square roots of the eigenvalues (i.e. the standard deviations of each principal component) are stored within the princomp object and can be extracted manually using `$sdev`. The principal component scores themselves can be accessed via `$scores` if these have been requested. There are also graphical methods associated with `princomp()` objects, `plot()` produces a screeplot, `biplot()` produces a biplot, we explain this tool further in section 9.8.4. However, we note an important aspect of eigenanalysis, which is very clearly stated in the helpfile and will be quite important later:

> The signs of the columns of the loadings and scores are arbitrary, and so may differ between different programs for PCA, and even between different builds of R.

This is a point we will return to a few times particularly when considering bootstrapping!

Executing a principal component analysis is therefore trivial, as demonstrated in the following code snippet. We extract the principal component scores and create qq plots; as demonstrated in chapter 2 any univariate linear combination of multivariate normal data should be normally distributed.

```
hept.princomp <- princomp(hept.df[,-1], scale = TRUE)
summary(hept.princomp)
plot(hept.princomp) ## produces scree plot
par(mfrow = c(3,3))
apply(hept.princomp$scores, 2, qqnorm)
```

9.8.4 Inbuilt functions 2: `prcomp()`

The preferred **R** function for a conventional principal component analysis is `prcomp()`. There are a number of differences in use and extraction, but the rather more important difference is that it is based upon a singular value decomposition of the data. The singular value decomposition was outlined in Section 2.6. Whilst discussing the singular value decomposition in this context it is convenient to introduce the biplot. Gabriel (1971) introduced the biplot as a means of representing either a matrix of rank two, or a rank two approximation to a matrix of rank greater than two. The idea is to display vectors for each row and vectors for each column are displayed on the same plot, illustrating features

of either a rank two matrix or a rank two approximation. Whilst we illustrate it in detail here, it has application in areas other than with principal components.

As discussed earlier, we consider data matrix $\boldsymbol{X}$; it's decomposition is given by:

$$\boldsymbol{X} = \sum_{i=1}^{p} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

and hence for the biplot, where we specifically require a rank two solution:

$$\boldsymbol{X} = \sum_{i=1}^{2} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T = (\boldsymbol{u}_1, \boldsymbol{u}_2) \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^T \boldsymbol{v}_2^T \end{pmatrix}$$

Based on this decomposition, we have a choice of plots. In general, we can plot:

$$\boldsymbol{g}_i^T = \lambda_1^{1-\zeta} u_{1i}, \lambda_1^{1-\zeta} u_{2i}$$

for the observations and

$$\boldsymbol{h}_j^T = \lambda_1^{\zeta} q_{1j} \lambda_2^{\zeta} q_{2j}$$

for the columns. where $0 \leq \zeta \leq 1$. Gabriel (1971) essentially gave proposals for $\zeta = 0$, $\zeta = 0.5$ and $\zeta = 1$. In **R**, these can be set with the argument `scale` which takes the range $0 \leq$ scale ≤ 1. The default is `scale = 1` which implies $\boldsymbol{H}^T \boldsymbol{H} = \boldsymbol{I}$; more notably this means that the inner products between variables approximate covariances and distances between observations approximate Mahalanobis distance. By default, observations are scaled up by $\sqrt{n}$, variables are scaled down by $\sqrt{n}$

It is possible to adjust `choices=c(1,2)` if you don't really want a biplot but want to examine the projection of what are in this context other principal components.

```
pairs(turtles[,-1],
   lower.panel = function(x, y){ points(x, y,
   pch = unclass(turtles[,1]),
   col = as.numeric(turtles[,1]))},
   main = "Pairwise scatter plots for painted turtles")
```

We follow Flury (1997) in transforming these data onto the log scale (and multiplying them by 10). This is quite common in allometric applications, where the $\log$ transformation may suffice in terms of bringing all variables onto a comparable scale.

```
data(turtles)
  turtles.m <- subset(turtles, turtles$Gender == "Male")
  turtles.m <- 10 * log(turtles.m[,-1])
  turtles.m.prcomp <- prcomp(turtles.m)
```

```
  summary(turtles.m.prcomp)
plot(turtles.m.prcomp)
turtles.m.prcomp$sdev^2 ## extract eigenvalues
par(xpd = NA)
biplot(turtles.m.prcomp)
```

We can illustrate the biplot as follows by carrying out the various computations by hand within **R**:

```
turtles.svd <- svd(turtles.m)
H <- cbind(turtles.svd$u[,1], turtles.svd$u[,2]) * sqrt(24)
g1 <- turtles.svd$d[1] * turtles.svd$v[,1] / sqrt(24)
g2 <- turtles.svd$d[2] * turtles.svd$v[,2] / sqrt(24)
plot(H)
par(new = TRUE)
plot(c(-arrx,arrx),c(-arry,arry), type = "n",
  xaxt = "n", yaxt = "n")
axis(3)
axis(4)
arrows(0, 0, arrx, arry)
```

9.9 Principal Components Regression

Finally, we say a few words about principal components regression, and illustrate the use of principal components in bioinformatics with the superpc package.

9.10 "Model" criticism for principal components analysis

This section has deliberately given a provocative title. "Model" criticism clearly requires some kind of model, it is worth giving some thought to whether we are using a principal components model in the context of a model or not. It is possible to use the technique as a data analytical technique, particularly when used for data reduction there is no necessity to assume multivariate normality. However, when we are using it in the context of a multivariate normal distribution, it is important to be aware of a number of key distributional results on the asymptotic distribution of the eigenvalues and eigenvectors of a covariance matrix.

9.10.1 Distribution theory for the Eigenvalues and Eigenvectors of a covariance matrix

Girshink (1939); Anderson (1963) give results for asymptotic distributions in connection with the covariance matrix. Firstly, we comment on the existence of maximum likelihood estimators for the eigendecomposition of a covariance matrix:

Theorem 9.11 *Under conditions of normality, the maximum likelihood estimators for the population eigenvalues and eigenvectors of* $\mathbf{\Sigma}$ *are given by the sample eigenvalues and eigenvectors, provided all the eigenvalues are distinct.*

Proof: See (Mardia et al., 1979, page 229) and (Anderson, 1984, page 460)

Theorem indicates that our sample eigenvalues are the maximum likelihood estimators of their corresponding population counterparts. In most inferential situations we would wish to qualify such an estimator with guidance as to the level of associated uncertainty. Firstly, we wish to establish the distribution of the eigenvalues.

Theorem 9.12 *The asymptotic distribution of eigenvalues and eigenvectors can be expressed as follows:*

$$\sqrt{n}(\boldsymbol{\ell} - \boldsymbol{\lambda}) \sim MVN(\mathbf{0}, 2\mathbf{\Lambda}^2) \tag{9.20}$$

where Λ *is the diagonal matrix of eigenvalues. This can be expressed equivalently as:*

$$\sqrt{n}\left(\log(\boldsymbol{\ell}) - \log(\boldsymbol{\lambda})\right) \sim MVN(\mathbf{0}, 2) \tag{9.21}$$

and

$$\sqrt{n}(\hat{\boldsymbol{e}}_i - \boldsymbol{e}_i) \sim MVN(\mathbf{0}, \boldsymbol{E}_i) \tag{9.22}$$

where $\boldsymbol{E}_i = \lambda_i \sum_{k=1, k\neq i}^{p} \frac{\lambda_k}{(\lambda_k - \lambda_i)^2} \boldsymbol{e}_k \boldsymbol{e}^T$

Proof: The proof for this has been restated in Flury (1988). These properties were established following work by Anderson (1963) and Girshink (1939).

Theorem 9.20 indicates that for large n, the eigenvalues λ_i are independently distributed. We can therefore obtain standard errors for the eigenvalues as follows:

$$se(\lambda_j) = \sqrt{2/n}\lambda_j,$$

giving confidence intervals:

$$\frac{\ell_i}{1+\sqrt{2/n}z_{\alpha/2}} \leq \lambda_i \leq \frac{\ell_i}{1+\sqrt{2/n}z_{1-\alpha/2}}$$

and the standard error for the corresponding eigenvectors are given by:

$$se(\alpha) = \left(\frac{1}{n}\lambda_j \sum_{j+1;j\neq h}^{p} \frac{\lambda_j}{(\lambda_j-\lambda_h)^2}\alpha_{jh}^2\right)^{1/2}.$$

We can illustrate calculation of these standard errors, as well as estimation of associated confidence intervals by adapting code written by Marco Bee to accompany Flury (1997)). This code will be set out at an S3 class in **R**. Firstly therefore, we set out a container:

```
lpc <- function(X){
  UseMethod("lpc", X)
}
```

And now we can write out a default method which calculates the relevant confidence intervals:

```
lpc.default <- function(X)
{
n <- dim(X)[1]; p <- dim(X)[2]  # number of observations
X.prcomp <- prcomp(X)
evals <- X.prcomp$sdev^2
Ones <- matrix(1, p, p)
Lambda <- Ones * evals
Q <- (t(Lambda) - Lambda + diag(p))^(-2) - diag(p) # nifty trick
Theta1 <- sweep(Q, 2, evals, FUN="*")
Theta <- Theta1 * evals # compute matrix of theta-coefficients
stdeB <- matrix(0,p,p)
   h <- 1
      while (h <= p){
      V <- X.prcomp$rotation %*%
          (Theta[, h] * t(X.prcomp$rotation))
      stdeB[, h] <- sqrt(diag(V)/n)
      h <- h + 1
      }
stdelam <- sqrt(2/n) * evals
results <- list("eigenvectors" = X.prcomp$rotation,
  "eigenvalues" = X.prcomp$sdev^2,
  "stdeB" = stdeB, "stdelam" = stdelam)
  class(results) <- "lpc"
  results
}
```

Having returned the standard error we can write a simpler print function which displays the eigenvectors and eigenvalues along with their associated standard error:

```
print.lpc <- function(x) {
print(x[1]) ## eigenvectors
print(x[2]) ## eigenvalues
cat("standard errors for eigenvector coefficients:")
print(x[3])
cat("standard errors for eigenvalues:")
print(x[4])
cat("\n\n")
invisible(x)
}
```

So for example, with the turtles data:

```
> lpc(subset(turtles, turtles$Gender == "Male")[,-1])
$eigenvectors
          [,1]        [,2]        [,3]
[1,] 0.8401219  0.48810477 -0.23653541
[2,] 0.4919082 -0.86938426 -0.04687583
[3,] 0.2285205  0.07697229  0.97049145

$eigenvalues
[1] 195.274633   3.688564   1.103833

standard errors for eigenvector coefficients:$stdeB
           [,1]       [,2]       [,3]
[1,] 0.01442666 0.04469703 0.07885419
[2,] 0.02487011 0.01592627 0.13874655
[3,] 0.01513963 0.15478836 0.01276277

standard errors for eigenvalues:$stdelam
[1] 56.370931  1.064797  0.318649
```

And if we wanted to estimate the confidence intervals we can write an associated `summary` method which will do the additional calculations and return the results. Note in the code below that we have allowed for a *Bonferroni* adjustment. If we wish to adjust for making m comparisons we can replace $z_{\frac{\alpha}{2}}$ with $z_{\frac{\alpha}{2m}}$

```
summary.lpc <- function(x, alpha = 0.05, bonferroni = FALSE) {
if (!is.null(alpha)){ ## calculate ci if asked
  if (bonferroni == TRUE) {alpha = alpha / length(x[[2]])}
  z <- abs(qnorm((1-alpha)/2))
```

```
}
print(x[1]) ## eigenvectors

if (!is.null(alpha)){
cat(round(alpha * 100), "\% CI: \n ")
veclo <- x[[1]] - z * x[[3]]
vechi <- x[[1]] + z * x[[3]]
print(veclo)
print(vechi)
cat("\n")
}

print(x[2]) ## eigenvalues

if (!is.null(alpha)){
cat(round(alpha * 100), "\% CI: \n ")
vallo <- x[[2]] - z * x[[4]]
valhi <- x[[2]] + z * x[[4]]
print(vallo)
print(valhi)
cat("\n")
}

cat("standard errors for eigenvector coefficients:")
print(x[3])

cat("standard errors for eigenvalues:")
print(x[4])
cat("\n\n")
invisible(x)
}
```

9.13 Sphericity

We preface this section on sphericity with some results concerning covariance / correlation matrices which are of less than full rank. If a symmetric positive definite matrix (correlation and covariance matrices are at least semi-definite). If such a matrix is of full rank p then all the eigen values are positive. If the rank of the covariance or correlation matrix $m < p$ then the last $p - m$ eigenvalues are identically zero. The converse of this theorem is that any non-zero eigen value can be considered to be *significantly* non-zero.

However, we are now going to consider sphericity, where there are not p distinct eigenvalues. We highlighted earlier the potential problem of sphericity and the effect on a resultant principal component analysis. Clearly there is little point carrying out a principal component analysis under conditions of sphericity. We can consider three possiblilites, where $\boldsymbol{R} = \boldsymbol{I}$, which can arise either because $\boldsymbol{S} = s\boldsymbol{I}$ or the more general possibility that $\boldsymbol{S}$ is diagonal.

We firstly consider the most general possibility, that $\boldsymbol{\Sigma} \propto \sigma \boldsymbol{I}$ where σ is unspecified. However, this test is equivalent to examining whether all the roots of $|\boldsymbol{\Sigma} - \lambda \boldsymbol{I}| = 0$ are equal. In this eventuality, the arithmetic and geometric means will be identical

We firstly consider a general test for sphericity proposed by Mauchly (1940)

Theorem 9.14 *We consider that:*

$$\frac{\prod_{j=1}^{p} \lambda_i^{1/j}}{\sum_{j=1}^{p} \lambda_i^{1/j}} = \frac{|\boldsymbol{\Sigma}|^{1/j}}{\frac{1}{j} tr(\boldsymbol{\Sigma})}$$

This yields the following test statistic. Under the null hypothesis $H_0 : \boldsymbol{S} = \sigma \boldsymbol{I}$, the test statistic m given by:

$$m = \frac{|\boldsymbol{S}\boldsymbol{\Sigma}_0^{-1}|^{n/2}}{\left[\frac{1}{p} trace(\boldsymbol{S}\boldsymbol{\Sigma}_0^{-1})^{pn/2}\right]} \tag{9.23}$$

This is pre-implemented in **R** for manova type objects, therefore if we fit a null manova object we can carry out this test:

```
obj <- manova(as.matrix(turtles.m) ~ 1)
mauchly.test(obj)

        Mauchly's test of sphericity

data:  SSD matrix from manova(as.matrix(turtles.m) ~ 1)
= 101.1821, p-value < 2.2e-16
```

And so we have evidence (provided the test assumptions are met) that the turtle data are not spherical.

For completeness, we mention here another test for sphericity is given by (Morrison, 2005, see section 1.9) based upon the determinant of the correlation matrix.

Definition 9.15 *Under the null hypothesis $H_0 : \boldsymbol{R} = \boldsymbol{I}$, the test statistic w given by:*

$$w = -\left(n - \frac{2p+5}{6}\right) \log|\boldsymbol{R}|$$

has a χ^2 distribution with $\frac{1}{2}p(p-1)$ degrees of freedom.

This test is quite simply coded up in **R**.

```
morrison <- function(data){
  n <- dim(data)[1]; p <- dim(data)[2];
  wnm <- -(n - (2 * p)/6) * log(det(cor(data)))
    cat(paste("wnm = ", wnm, "\n"))
    cat(paste("df = ", p * (p - 1) * 0.5, "\n"))
    cat(paste("Chisq density = ", dchisq(wnm, p * (p - 1) * 0.5) , "\n"))
}
```

Again, this test confirms non-sphericity of the Turtles data.

9.15.1 Partial sphericity

It is usually the case that partial sphericity is or more practical concern than more complete independence of the data. We will consider more heuristic methods for selecting the dimensionality of a principal component projection later, but clearly the eigenvectors of principal components with equal eigenvalues are too poorly defined to be of any practical use. We are therefore interested in partial sphericity, where $\lambda_{q+1} = \lambda_{q+2} = \ldots = \lambda_p$

Where we have partial sphericity, we may note the following theorem:

Theorem 9.16 *For normal data, where the eigenvalues of $\mathbf{\Sigma}$ are not distinct then the m.l.e. of $\bar{\lambda}$ is the corresponding arithmetic mean of the sample eigenvalues, and the corresponding eigenvectors are maximum likelihood estimators although they are not unique.*

Proof: See Anderson (1963)

The asymptotic theory set out above leads to a number of possible tests for partial sphericity. One likelihood ratio can be considered as follows:

Definition 9.17 *(Seber, 1984, page 198) gives a likelihood ratio test for partial sphericity. In order to determine whether whether the last $p - q$ eigenvalues are equal can be specified as follows. We wish to test $H_0 : \lambda_{q+1} = \lambda_{q+2} = \ldots = \lambda_p$ versus $H_a : \lambda_{q+1} > \lambda_{q+2} > \ldots > \lambda_p$.*

$$-2\log = -n \log \left(\prod_{q=1}^{p} \frac{\lambda_j}{\bar{\lambda}} \right) \tag{9.24}$$

where $\bar{lambda} = \frac{1}{p-q} \sum_{j=q}^{p} \lambda_j$. Under the null hypothesis, the likelihood ratio statistic, $-2\log$, of the eivenvalues derived from a covariance matrix is distributed as $\chi^2_{\frac{1}{2}(p-q-1)(p-q+2)}$. The test can be applied to the correlation matrix but the asymptotic distribution is no longer chi-square. The asymptotics can be improved with a correction proposed by Lawley (1956):

$$-\left\{n-1-q-\frac{2(p-q)^2+(p-q)+2}{6(p-q)}+\sum_{j=1}^{q}\left(\frac{\bar{\lambda}}{\lambda_j-\bar{\lambda}}\right)^2\right\}\log\left(\prod_{q=1}^{p}\frac{\lambda_j}{\bar{\lambda}}\right) \tag{9.25}$$

This test is however not robust to departures from normality.

It is reasonably straightforward to start coding a function to execute this in **R**, a sketch of such a function is illustrated here:

```
spher <- function(X, q){
  p <- dim(X)[2]; n <- dim(X)[1]; r <- p-q
  X.prcomp <- prcomp(X)
  evals <- X.prcomp$sdev^2
  retain <- evals[1:q]; discard <- evals[-(1:q)]
  lambdahat <- mean(discard)
  bit <- sum(lambdahat / (retain - lambdahat) )^2
  corr <- n - 1 - q - (2 * r^2 + r + 2)/(6*r) + bit
  product <- prod(discard / lambdahat)
  lrt <- -corr * log(product)
  df <- 0.5 * (r-1) * (r+2)
    cat(paste("-2log L = ", lrt, "\n") )
    cat(paste("df = ", df, "\n") )
    cat(paste("Chisq density ", dchisq(lrt, df ), "\n" ))
    ##return(lrt)
}
```

We illustrate this with the turtle data. Recalling that the first eigenvalue was 2.33, considerably greater than the second and third eigenvalues of 0.06 and 0.036.

```
> spher(turtles.m, 1)
-2log L =  1.34290454195737
df =  2
Chisq density  0.255482988814162
```

So in this case we cannot reject H_0 and have no evidence that the second and third eigenvalues are distinct. We might be included to consider the possibility here that that $\lambda_2 = \lambda_3$ and would therefore be somewhat wary of the resultant eigenvectors.

A simpler explanation is given in (Flury, 1997, page 622), which follows from that given in (Anderson, 1984, page 475), and is given in slightly different form in (Mardia et al., 1979, page 235).This relies on a log likelihood statistic derived as a ratio of arithmetic to geometric means of the eigenvalues.

```
function(X, q){
  p <- dim(X)[2];  n <- dim(X)[1];  r <- p-q
```

```
  X.prcomp <- prcomp(X)
  evals <- X.prcomp$sdev^2
  q1 <- q+1
    discard <- evals[q1:p]
    a <- sum(discard) / r
    g <- prod(discard)^(1/r)
    s <- n * r * log(a / g)
    df <- 0.5 * r * (r+1)
  cat(paste("-log L = ", s, "\n") )
  cat(paste("df = ", df, "\n") )
  cat(paste("Chisq density ", dchisq(lrt, df ), "\n" ))
  ##return(s)
}
```

Asymptotically, this value can be considered to follow a $\chi^2_{\frac{1}{2}(p-q)(p-q+1)-2}$ distribution under the null hypothesis. It can be seen that is related to the test of complete sphericity. This test can be used with the turtle data and again confirms the possible that the second and third principal components can be considered spherical and should not be interepreted further.

High Dimensional Tests for Sphericity

Having (hopefully) demonstrated the importance of sphericity in conventional principal components analysis we refer to results which carry out analagous procedures in high dimensions. Whilst the asymptotic tests above rely on $n \rightarrow \infty$, it can be problematic where $p > n$. More recent work therefore examines how one might carry out tests for this eventually. We firstly consider testing whether $\boldsymbol{\Sigma} \propto \boldsymbol{I}$

Theorem 9.18 *A test for $\boldsymbol{\Sigma} \propto \boldsymbol{I}$ which is reliable whenever $p > n$ can be given as follows:*

$$U = \frac{1}{p} trace\left((\frac{\boldsymbol{S}}{(1/p)trace(\boldsymbol{S})} - \boldsymbol{I})^2 \right) \tag{9.26}$$

In this case, it may be noted that $\frac{np}{2}U$ asymptotically follows a chi^2 distribution with $\frac{1}{2}p(p+1)-1$ degrees of freedom.

Proof: See Ledoit and Wolf (2002), following work by John (1971, 1972)

This can be very simply estimated in **R**:

```
JohnsU <- function(data){
  p <- dim(data)[2]
  S <- cov(data)
  traceS <- sum(diag(S))
```

```
  traceSI <- sum(diag(S-diag(rep(1, p))))
    u <- 1/p * traceS / (1/p*traceSI^2)
    test <- n * p * 0.5 * u
    df <- (0.5 * p * (p+1)) - 1
  cat(paste("U = ", test, "\n") )
  cat(paste("df = ", df, "\n") )
  cat(paste("Chisq density ", dchisq(test, df ), "\n" ))
}
```

So we can estimate the sphericity quite simply

```
JohnsU(khan$train)
```

Which appears to indicate little evidence for sphericity.

Testing the correlation matrix is not quite so straighforward to expansion of p relative to n. $\boldsymbol{\Sigma} = \boldsymbol{I}$

Theorem 9.19 *A test for $\boldsymbol{\Sigma} = \boldsymbol{I}$ which is reliable whenever $p > n$ is given by:*

$$W = \frac{1}{p} trace\left\{(\boldsymbol{S} - \boldsymbol{I})^2\right\} - \frac{p}{n}\left\{\frac{1}{p} trace(\boldsymbol{S})\right\}^2 + \frac{p}{n} \tag{9.27}$$

Under H_0, assuming multivariate normality $\frac{nm}{2} W \rightarrow^d \chi^2_{p(p+1)/2-1}$

Proof: Ledoit and Wolf (2002)

```
ledoitwolf <- function(data){
  n <- dim(data)[1]; p <- dim(data)[2]
  S <- cor(data)
  traceS <- sum(diag(S))
  SI <- crossprod(S - diag(rep(1, p)))
  traceSI <- sum(diag(SI))
    w <- 1/p*traceSI - p/n*(1/p*traceS)^2 + p/n
    test <- n * p * 0.5 * w
    df <- (0.5 * p * (p+1))
  cat(paste("U = ", test, "\n") )
  cat(paste("df = ", df, "\n") )
  cat(paste("Chisq density ", dchisq(test, df ), "\n" ))
}
```

However, we can consider results based on those indicated in 9.15 which can be used to test whether $\boldsymbol{R} = \boldsymbol{I}$.

Theorem 9.20 *A test statistic for sphericity is given by:*

$$t = \sum_{i=2}^{p} \sum_{j=1}^{i-1} r_{ij}^2 - \frac{p(p-1)}{2n}$$

Which is asymptotically normal with zero mean and variance:

$$\sigma_t^2 = \frac{p(p-1)(n-1)}{n^2(n+2)}$$

Proof: See Schott (2005)

This is simply illustrated in **R**.

```
schott <- function(data){
  n <- dim(data)[1]; p <- dim(data)[2]
  R <- cor(data)
  R[lower.tri(R) == FALSE] <- NA
  red <- na.omit(as.vector(R))
   tnm <- sum(red^2)
   cf <- (p * (p-1)) / (2 * n)
   test <- tnm - cf
   sigma2 <- (p * (p-1) * (n-1)) / (n^2 * (n+2) )
  cat(paste("tnm = ", tnm, "cf = ", cf,  "\n") )
  cat(paste("Normal density ", dnorm(test, sqrt(sigma2) ), "\n" ))
}
```

Again, a call to `schott(khan$train)` confirms that these data are non-spherical.

As before, perhaps we are more interested in the generalisations of these statistics, i.e. we are concerned with partial sphericity and wish to determine whether the smallest $p - q$ eigenvalues of $\mathbf{\Sigma}$ are equal.

Theorem 9.21 *Generalising equation 9.27, a test for partial sphericity can be based upon the following test statistic:*

$$u = \frac{(1/p)\sum_{i=q+1}^{p} \lambda_i}{\left[(1/p)\sum_{i=q+1}^{p} \lambda_i\right]^2} - 1 \tag{9.28}$$

where $\frac{n-q}{u}$[1] *can be compared with a* χ^2 *distribution with* $p(p+1)/2 - 1$ *degrees of freedom.*

Proof: Schott (2006)

Again, this test can be coded up in **R** and examined in the context of the khan data.

[1] check this

```
schottpartial <- function(X, q){
  p <- dim(X)[2]; n <- dim(X)[1]; r <- p-q
  X.prcomp <- prcomp(X)
  evals <- X.prcomp$sdev^2
  discard <- evals[-(1:q)]
  u <- (sum(discard^2) / r) / (sum(discard) / r)^2  - 1
  df <- 0.5 * r * (r+1) - 1
    cat(paste("u = ", u, "\n") )
    cat(paste("df = ", df, "\n") )
    cat(paste("Chisq density ", dchisq(u * (n-q), df ), "\n" ))
    ##return(lrt)
}
```

It is bearing in mind that just because the smallest $p - q$ eigenvalues indicate the corresponding principal components explain very little of the variation, they do not necessarily contain any useful information.

9.22 How many components to retain

We have considered formal hypothesis testing for sphericity and partial sphericity. This may well indicate that it is not sensible to include a number of principal components in a given representation of multivariate data. However, we may not really be interested in modelling multivariate normality. We may not like asymptotics. We will consider a few further results on selecting the number of principal components in any given projection of the data.

Whilst principal components have optimality properties in terms of providing the best lower dimensional projection in terms of mean squared error. Nevertheless, they are often used, however informally, in an inferential role. Consideble care is needed in their interpretation. It is important to note that we are working with sample principal components, these can be somewhat unstable relative to the puted underlying population components. It makes little sense to attempt anything resembling inferential work without considering the stability of a particular principal component solution. Typically, having decided how best to scale the data, the next most important question surrounds how many components need to be retained. We will first consider some of the more informal procedures used to guide this judgement, and will subsequently consider methods derived from normal theory inference.

9.22.1 Data analytic diagnostics

A number of proposals have been made in the literature concerning decisions surrounding the "number of components" to retain. The following are some of the more popular proposals:

Proportion of variance explained

The proportion of variance explained is a rather informal method for selecting q, the number of dimensions in the principal component projection required to adequately explain the data. Essentially, one decides *a priori* that a certain amount of variance is to be explained, and only accepts solutions meeting that requirement. It is however consistent with the exploratory nature to which principal component analysis is often applied.

We can however, using the asymptotic theory set out above, develop a confidence interval for the proportion of variance explained.

Theorem 9.23 *We denote our estimate of the proportion of variation explained by π:*

$$\pi = f(\lambda) = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i}.$$

If we also consider the corresponding sum of squares:

$$\zeta = \frac{\sum_{i=1}^{q} \lambda_i^2}{\sum_{i=1}^{p} \lambda_i^2}$$

Under conditions of multivariate normality, we can obtain an estimate of the variance associated π, the proportion of variance explained as follows:

$$\eta^2 = \frac{2trace(\Sigma)}{(n-1)(trace(\Sigma))^2} = \pi^2 - 2\zeta\pi + \zeta^2 \tag{9.29}$$

Proof: Sugiyama and Tong (1976) and (**?**, page 454) Hence we can derive a confidence interval for π as follows.

This can be illustrated as follows:

```
vals <-  hep.ev$values^2
q <- 3
alpha <- 0.95
alpha <- 1 - (1-alpha)/2

pi <- sum(vals[1:q]) / sum(vals)
alpha <- sum(vals[1:q]^2) / sum(vals^2)## by vector recycing
eta2 <- pi^{2} - 2 * alpha * pi + alpha^2
cat(pi, eta2)
cat("\n")
cat(pi + qnorm(alpha)*eta2)
cat("\n")
```

```
cat(pi - qnorm(alpha) * eta2)
cat("\n")
```

Change in slope of the scree plot

Cattell (1966) proposed the scree plot in the context of (principal component extracted) factor analysis. Without wishing to add to the confusion between the two techniques, it has become a fairly standard technique for assessing the adequacy of a number of dimension reducing techniques. Both `prcomp` and `princomp` objects have a plot method which yields a scree plot, the idea is to select components up to the point where the slope changes direction.

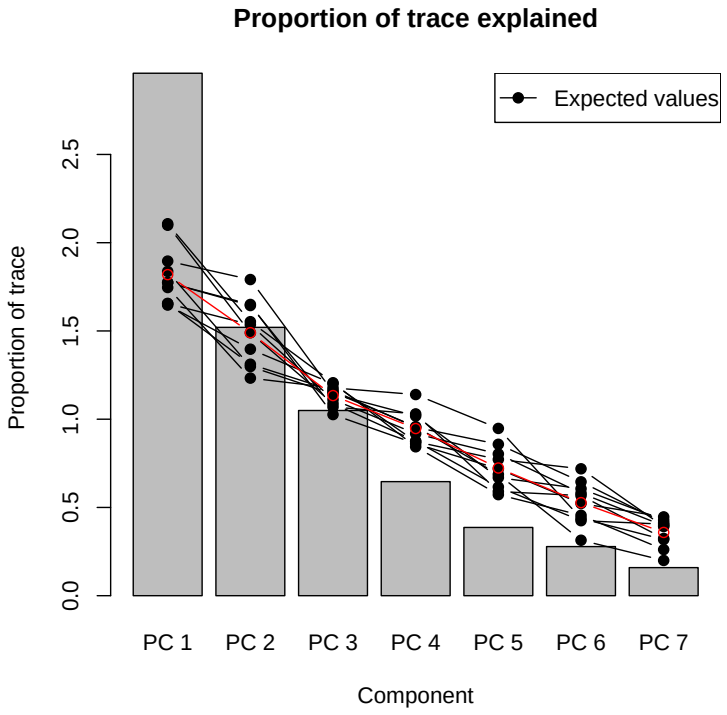
```
plot(hept.princomp)
```

Interpreting the scree plot in the presence of simulations

It is possible to extend the basic scree plot idea. Horn (1965) suggested simulating data from a multivariate normal having the same sample size, the same number of variables, the same means and variances but having zero covariances. There are a couple of manifestations of this approach within various **R** packages, for example `psy` package contains a ready made function. The scree plot of with zero correlation is expected to be a straight line, it can be compared with the scree plot from the observed data. It is possible to extend this to a full Monte Carlo test, the code listing below goes someway towards this.

```
Horn <- function(data, reps){
p <- dim(data)[2]
n <- dim(data)[1]
Varmat <- matrix(0,p,p)
Mean <- mean(data)
diag(Varmat) <- diag(var(data))

Evals <- princomp(data, cor = TRUE)$sdev^2
idx <- barplot(Evals, names.arg = paste("PC", c(1:7)),
xlab = "Component", ylab = "Proportion of trace",
main = "Proportion of trace explained")

results <- matrix(0,reps,p)
  for (i in 1:reps){
  SimData <- mvrnorm(n, Mean, Varmat)
  ExpEvalsH <- princomp(SimData, cor = TRUE)$sdev^2
  results[i,] <- ExpEvalsH
  lines(idx, ExpEvalsH, type = "b", pch = 16)
  }
```

```
lines(idx, apply(results, 2, mean), type = "b", col = "red")

legend("topright", lty = 1, pch = 16, legend = "Expected values")
Results <- data.frame(Evals = Evals, ExpEvalsH = ExpEvalsH)
}

Horn(hept.df[-1], 10)
```



Broken Stick

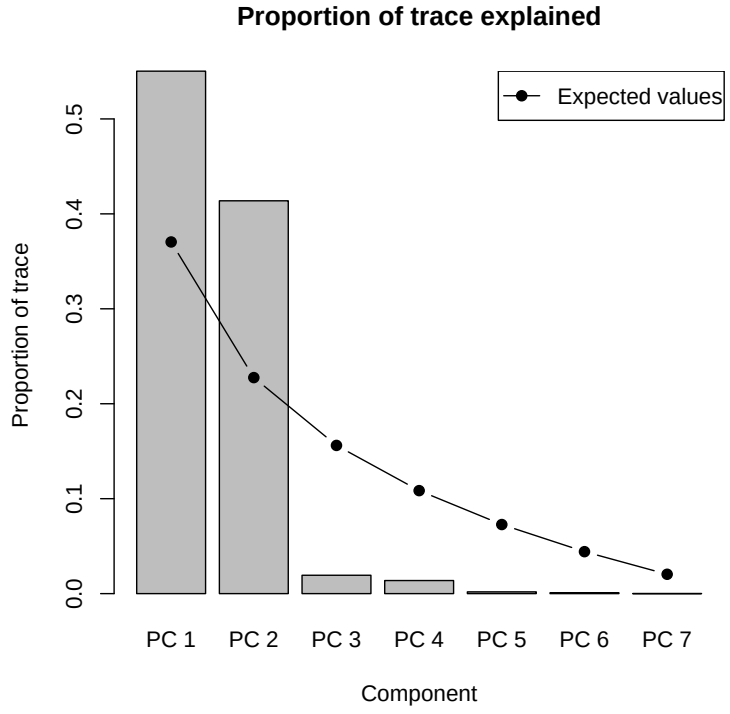
Another approach to assessing the proportion of variation explained has been made by Jolliffe (1986) who suggests using a "broken stick" approach. The idea here is that if any unit is randomly divided into p segments, the expected length of the kth longest segment is:

$$l_k = \left(\frac{1}{p}\right)\sum_{i=k}^{p}\left(\frac{1}{i}\right) \tag{9.30}$$

If we assume that the total variance, $trace\boldsymbol{S} = \sum_{j=1}^{p}$, is the "stick", we can use this approach to estimate an expected size of each eigenvalue. A rather simple function to calculate the expected values indicated by 9.30 is given below, the expected values are plotted alongside the observed values from a `princomp()` object created from the Heptathalon data. It should be noted that `princomp()`

returns the standard deviations (`$sdev`), these have therefore been squared to recoved the eigenvalues λ_i.

```
> stickometer <- function(p){
  vec <- 1 / (1:p)
  stick <- vector("numeric", p)
  stick[1] <- sum(vec)
     for (i in 2:p){
     stick[i] <- sum(vec[-(1:(i-1))])}
  stick <- 1/p * stick
  names(stick) <- paste("Comp.", c(1:p), sep = "")
  return(stick)
}
>
> stick <- stickometer(7)
> proptrace <- hep.princomp$sdev^2 / sum(hep.princomp$sdev^2)
>
> stick
> proptrace
>
> idx <- barplot(proptrace, names.arg = paste("PC", c(1:7)),
> xlab = "Component", ylab = "Proportion of trace",
> main = "Proportion of trace explained")
> lines(idx, stick, type = "b", pch = 16)
> legend("topright", lty = 1, pch = 16, legend = "Expected values")
```



Examination of the results (numbers or the plot) suggest that the first and seventh components are accounting for slightly more variation than would be expected purely by chance.

Kaiser Criterion

The Kaiser Criteria is a rather inflexible criteria widely met in many sofware packages. Basically it amounts to retaining all components where the eigenvalue is greater than the mean of the eigenvalues. In the case of principal components based on the correlation matrix this clearly means retaining all components where the eigenvalue is greater than one. Whilst one may not wish to assume multivariate normality, the asymptotics considered next provide a clear warning that population eigenvalues greater than one could clearly be realised in a sample with values below one.

Karlis provide some caveats on the use of this criteriosn.

9.23.1 Cross validation

Cross-validation in the context of principal components was proposed by Wold (1976, 1978) and developed by Eastment and Krzanowski (1982). In essence, the sample can be randomly split into g groups, the loadings can be estimated from reduced sets omitting each of the g groups in turn, but the predicted values can be found from these g groups using the loadings estimated from the other rows. The value for $(\hat{\boldsymbol{x}} - \boldsymbol{x})$ can be estimated from equation 9.32, and the PRESS statistic estimated.

```
pcaxv <- function(X){
  UseMethod("pcaxv", X)
}
```

It is useful to create an S3 object to carry out this procedure. The working function is given by:

```
pcaxv.default <- function(X, g = 5){
    N <- dim(X)[1]
    p <- dim(X)[2]
      index <- sample(c(1:N) )
      groups <- gl(g, N %/% g)
         Q <- matrix(0, g, p)
           for (i in 1:g){
           dot <- prcomp(X[-index[groups == i],])
         Q[i,] <- colSums((as.matrix(scale(X))[index[groups == i],]
                       %*% dot$rotation)^2)}
    colmeans <- colSums(Q) / N
  PRESS <- cumsum(colmeans[c(p:1)])/ c(1:p)
  PRESS <- PRESS[c(p:1)]
```

```
  names(PRESS) <- paste("C", c(0:(p-1)))
  results <- list("PRESS" = PRESS,
    dm = pcaxvconstants(N,p)$dm, dr = pcaxvconstants(N,p)$dr)
class(results) <- "pcaxv"
results
}
```

The function `pcaxvconstants()` calculates some constants that can be used in the summary function. A suitable print function for use with cross-validation objects created here can be given as follows:

```
print.pcaxv <- function(x){
cat("Components Removed \n")
  print(x[[1]])
  cat("\n")
  invisible(x)
}
```

(Jackson, 1991, page 354) refers to a W statistic (without giving any idea as to its origin or distribution. The idea behind this W statistic however is that for any component where $W > 1$ we have evidence to retain the component, where $W < 1$ we have an adequate representation of our data swarm without that component. The constants calculated earlier are basically $D_M = n+p-2(p-q)$, $D_R = p(n-1) - \sum_{i=1}^{(p-q)}(n+p-2i)$, and W is given by:

$$W = \frac{(PRESS((p-q)-1) - PRESS(p-q))/D_M(p-q)}{PRESS(p-q)/D_R(p-q)} \tag{9.31}$$

```
summary.pcaxv <- function(x){
  cat("PRESS for components Removed \n")
  print(x[[1]])
  cat("\n")
    wl <- length(x$PRESS)-1
    w <- rep(NA, wl)
       for (i in 1:wl){
       w[i] <- ((x$PRESS[i] - x$PRESS[i+1]) / x$dm[i+1] ) /
             (x$PRESS[i+1] / x$dr[i+1] ) }
    names(w) <- paste("C", c(1:wl))
  cat("W for components included \n")
  print(w)
invisible(x)
}
```

These are rather interesting concepts in practice. For example, considering the `turtle` data examined earlier:

```
> turtle.xv <- pcaxv(as.matrix(log(turtles[,-1])))
> turtle.xv
PRESS for components Removed
       C 0        C 1        C 2
0.97916667 0.05952219 0.06118754

W for components included
        C 1         C 2
29.00900476 -0.02605895
```

Which appears to provide strong evidence that the turtle data can be represented by one principal component.

One little glurp can happen with the W statistic. Consider the water strider data given in Flury (1997).

```
data(strider)
dot <- pcaxv(as.matrix(log(strider)))
summary(dot)
PRESS for components Removed
       C 0        C 1        C 2        C 3        C 4        C 5
0.98863636 0.27912037 0.23714880 0.15109046 0.10068831 0.05974256

W for components included
       C 1        C 2        C 3        C 4        C 5
11.8809534  0.6686066  1.6310744  0.9662281  0.6690516
```

It can be seen that the second component has a W below 1, but for the third is clearly above 1, and for the fourth is is very close to 1. Jackson (1991) suggests that this may be due to the presence of outliers - this is left as an exercise for further examination.

The following functions (a) need tidying up and (b) support the pcaxv function - they're left here for tidiness only;

```
rtota <- function(N, p, q){
  rtot <- 0
  for (i in 1:q){
  rtot <- rtot + N + p - 2 * i
  }
rtot
}
```

```
pcaxvconstants <- function(N,p){
  dm <- N + p - 2 * (p - c(p:1))
  dm[1] <- NA
    dr <- rep(0,p)
```

```
    dr[1] <- p * (N-1)
  for (i in 2:p){
  dr[i] <- p * (N - 1) - rtota(N, p, i-1)
}
results <- list(dm = dm, dr = dr)
}
```

Bootstrapping

We discuss standard errors derived from asymptotic theory in the next section, but clearly there are limitations in having to assume multivariate normality. Bootstrapping avoids any such assumptions, we can make inference based upon an empirical distribution function. For tidiness, we will make a little function that calls `prcomp()` and returns the eigenvalues and eigenvectors only:

```
theta <- function(x.data, x){
eta <- prcomp(x.data[x,])
return(cbind(eta[[1]], eta[[2]]))
}
```

However, whilst computer power might be cheap, nothing in life is free and the problem with eigenvectors is the arbitrariness of the signs. Accordingly, it is completely unacceptable to use bootstrapping without checking for inversions of eigenvectors. Below we consider carrying out some boostrapping, plot the results and identify the most unreasonably volatile eigenvector. We can use the sign of this eigenvector to adjust the signs of all the other components of this eigenvector and hence obtain bootstrap estimates of the eigenvectors.

Then we call the function with our data, and tell it how many sets of bootstraps we want:

```
> library(boot)
> hep.boot <- boot(hep.scale, theta, R = 50, sim = "ordinary")
> eigen.bootplot(hep.boot,8,7)
```

It is quite clear that we need to invert some eigenvectors. One approach is to identify one coefficient within an eigenvector and whenever this is below zero to multiply the entire vector by the scalar -1.

```
> idx <- hep.boot$t[,8] < 0
> hep.boot$t[idx,c(8:14)] <- hep.boot$t[idx,c(8:14)] * -1
> eigen.bootplot(hep.boot,8,7)
```

In this way we can obtain additional information on the variability of our principal component analysis.

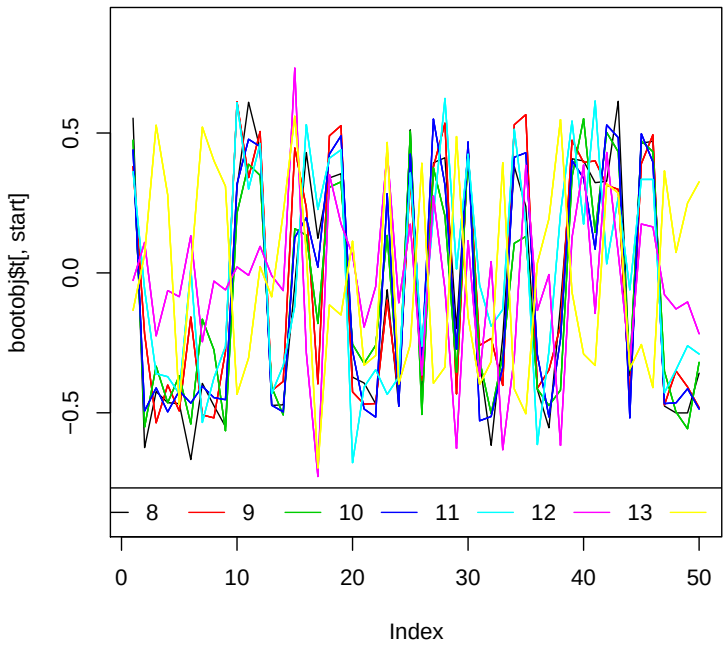


Figure 9.4: Trace of bootstrap iterations for first eigenvector

Other structure matrices

We might also be interested in testing for other structure matrices, such as equi-correlation. See Lawley (1963).

$$\bar{r}_k = \frac{1}{p-1} \sum_{i=1;i\neq k}^{p} r_{ik}; k = 1, \ldots, p$$
$$\bar{r} = \frac{2}{p(p-1)} \sum_{i<k} \sum r_{ik}$$
$$\hat{alpha} = \frac{(p-1)^2 \left[1 - (1-\bar{r})^2\right]}{p - (p-2)(1-\bar{r})^2}$$

where $\frac{n-1}{(1-\bar{r})^2}\alpha$ follows a T^2 distribution.

Proof: JW 489

```
mean(spot[lower.tri(spot)])
```

9.23.2 Forward search

A more recent proposal for assessing the stability of principal component solution is the forward search Atkinson et al. (2004).

9.23.3 Assessing multivariate normality

If we are prepared to entertain the idea that our data might be multivariate normal, it is very simple to obtain distance measures from the principal component scores and examine the adequacy of a representation in this context (clearly this might not be so useful if we are cannot assume multivariate normality). Given a random vector $\boldsymbol{x}$ having mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, (Flury, 1997, pg 608-609) (page 608-609) demonstrates the relationship between the principal component scores z_j and the mahalanobis distance $\delta^2(\boldsymbol{x}, \boldsymbol{\mu})$ (which he calls the squared standard distance).

Theorem 9.24

$$\delta^2(\boldsymbol{x}, \boldsymbol{\mu}) = (\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) = \sum_{j=1}^{p} \frac{z_j^2}{\lambda_j} \tag{9.32}$$

where $\boldsymbol{z} = (z_1, \ldots, z_p)^T = \boldsymbol{E}(\boldsymbol{x} - \boldsymbol{\mu})$, *and* $\boldsymbol{\Sigma} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^T$ *as above.*

Proof: See Flury (1997)

We can use our principal component representation to partition the mahalanobis distance. We want $\delta_a^2 = \sum_{j=1}^{q} \frac{z_j^2}{\lambda_j}$ corresponding to the distance encapsulated in our first q principal components, and $\delta_b^2 = \sum_{j=q+1}^{p} \frac{z_j^2}{\lambda_j}$ corresponding to the distances encapsulated by the last $(p-q)$ principal components. Flury (1997) indicates that these can be represented by χ^2 random variables with respectively q and $p-q$ degrees of freedom which lends itself to diagnostic assesment of the adequacy of the principal component representation. It should perhaps be noted that this is a large sample approximation, Gnanadesikan (1977) (page 172) suggests $n = 25$ is adequate in the bivariate case. Bilodeau and Brenner (1999) (page 186) therefore indicate the use of the Beta distribution, with $\alpha = \frac{p-2}{2p}$ and $\beta = \frac{n-p-2}{2(n-p-1)}$.

It is possible to write a simple function to extract the distances associated with accepted and rejected principal component (and the total distance) which can then be used in various diagnostic plots.

```
> princomp2dist <- function(obj.princomp, retain){
 scores <- t(t(obj.princomp$scores^2) / obj.princomp$sdev)
 dtot <- apply(scores, 1, sum)
 d1 <- apply(scores[,c(1:retain)], 1, sum)
 d2 <- apply(scores[,-c(1:retain)], 1, sum)
 dists <- data.frame(dtot = dtot, d1 = d1, d2 = d2)
 return(dists)
}

> hept.princomp <- princomp(hept.df[-1], scores = TRUE, scale = TRUE)
> ## form a princomp object
> hept.m <- princomp2dist(hept.princomp, 3)
> ## extract distances based on 3 component representation
```

Having obtained the distances, we only need some suitable method of investigation. The most useful are qq-plots. Given we have only 26 rows and 9 columns, we will use a modified verion of the qqbeta function given by Bilodeau and Brenner (1999). This plots the Mahalanobis distance against a suitable beta distribution.

```
> qqbetaM <- function(x, p) {
  n <- length(x)
  a <- p/2
  b <- (n-p-1)/2
  alpha <- 0.5*(a-1)/a
  beta <- 0.5*(b-1)/b
  x <- sort(x)
  y <- qbeta(((1:n)-alpha)/(n-alpha-beta+1),a,b)*(n-1)^2/n
  plot(x,y,xlab="Mahalanobis distances",ylab="Beta quantiles")
}
> qqbetaM(hept.m$dtot,  7)
```

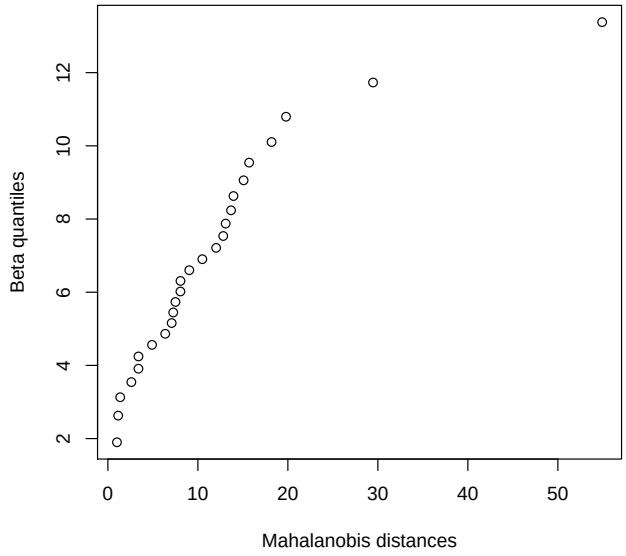


Figure 9.5: Mahalanobis distance from three component representation of the Heptathalon data versus theoretical quantiles of the Beta(1.5, 10) distribution

It is reasonably clear from figure 9.5 that there are reasons to doubt multivariate normality, particularly in relation to outliers.

Nevertheless, if we persevere, the adequacy of the $q = 3$ dimensional representation can be considered. Plotting δ_a^2 against δ_b^2 provides one way of identifying those points not well represented by the three dimensional projection.

```
> plot(hept.m$d1, hept.m$d2,
  xlab = "Represented by q", ylab = "Not represented by q",
  main = "Mahalanobis distances")
> identify(hept.m$d1, hept.m$d2, row.names(hept.df))
```

However, qq plots of the data do tend to suggest that she could be considered an outlier. This takes us back to the start of the chapter; in this case we may wish to consider a robust principal component analysis.

```
> hep.princomp <- princomp(hept.df[-1], cor = TRUE)
> hep.cor.rob <- cov.rob(hept.df[,-1], cor = TRUE)$cor
> hep.princomp.rob <- princomp(cov = hep.cor.rob)
> hep.princomp
> hep.princomp.rob
```

```
> loadings(hep.princomp)
> loadings(hep.princomp.rob)
```

In this case it should be seen that there is only a slight difference between the estimates for the eigenvalues, but the loadings do alter somewhat. Further methods for robust principal components will be considered in the next chapter.

9.25 Interpreting the principal components

This is the area that gets principal components a bad name. Sometimes referred to as *reification*, we look at the loadings on the various components, and try to suggest a concept that the component may be referring to. Factor analysis does something similar.

It is worth at this point considering the correlation between a given variable and the principal component score (or projection of the data in q dimensional subspace).

Definition 9.26 *The univariate correlation between a variable and it's principal compnent projection can be given by:*

$$\rho_{z,x_k} = \frac{e_i, \lambda_i}{\sqrt{\sigma_{kk}}}$$

Proof: (Johnson and Wichern, 1998, page 462)

It should be noted that this only measures the unvariate contribution of x to z, something Rencher (2002) feels is useless but something which may serve as a means to an end.

The corresponding measure for principal component analysis based on the correlation matrix is given by: $\rho_{z,x_{standardised}} = e_{ik}\sqrt{\lambda_i}$

9.27 Exercises

1. Consider data $\boldsymbol{X} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$. Find the covariance and correlation matrix for these data.
2. Consider $\boldsymbol{S} = \begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}$. Convert $\boldsymbol{S}$ into $\boldsymbol{R}$. Calculate the principal components from each matrix. Comare and contrast.
3. Calculate $\rho_{z,x}$
4. S = diag(2,4,2), what are evals and evecs

5. Couple of questions on equicorrelation matrices

6. Estimate Sx and Sx.

7. Carapace data - how many pcs (sphericity test, bootstrap, scree, blah blah blah)

Chapter 10

Canonical Correlation

10.1 Canonical variates

10.2 Interpretation

In canonical correlation, we are interested in the relationship between two sets of variables. We do this by creating linear combinations $\boldsymbol{U = a_1x_1 + a_2x_2 + \cdots + a_px_p}$ and $\boldsymbol{V = b_1y_1 + b_2y_2 + \cdots + b_qy_q}$ such that the correlation between $\boldsymbol{U}$ and $\boldsymbol{V}$ is as high as possible.

To do this, we need to work out the correlation matrix, and partition it:

$$
\begin{array}{c}
 \\
x_1 \\
\vdots \\
x_p \\
y_1 \\
\vdots \\
y_3
\end{array}
\begin{array}{c}
\begin{array}{cccccc} x_1 & \cdots & x_p & y_1 & \cdots & y_q \end{array} \\
\left(\begin{array}{c|c} A_{p\times p} & C_{p\times q} \\ \hline C_{q\times p} & B_{q\times q} \end{array}\right)
\end{array}
$$

Having done this, we calculate the matrix:

$$\boldsymbol{B^{-1}C^TA^{-1}C}$$

and find the associated eigenvalues (in descending order) $\lambda_1 > \lambda_2 > \ldots > \lambda_r$. The corresponding eigenvectors $\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_r}$ give the coefficients of the Y variables.

So:

$$\boldsymbol{v_i} = \boldsymbol{b_i^T Y}$$

where $\boldsymbol{b_i} = \begin{pmatrix} b_{i1} \\ \vdots \\ b_{iq} \end{pmatrix}$ and $\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{y_1} \\ \vdots \\ \boldsymbol{y_q} \end{pmatrix}$, or in longhand:

$$\boldsymbol{v_i} = b_{i1}\boldsymbol{y_1} + \cdots + b_{iq}\boldsymbol{y_q}$$

Having calculated these, it is possible to solve the coefficients for the X variables:

$a_1 = \boldsymbol{A^{-1}Cb_1}, a_2 = \boldsymbol{A^{-1}Cb_2}, \ldots, a_r = \boldsymbol{A^{-1}Cb_r}$,

f

$$\boldsymbol{u_i} = \boldsymbol{a_i^T X}$$

where $\boldsymbol{a_i} = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{iq} \end{pmatrix}$ and $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_r} \end{pmatrix}$, or in longhand:

$$\boldsymbol{u_i} = a_{i1}\boldsymbol{x_1} + \ldots + a_{ir}\boldsymbol{x_r}$$

And one really cute result is that $[corr(\boldsymbol{u_i}, \boldsymbol{v_i})]^2 = \lambda_i$.

10.3 Computer example

Franco Modigliani proposed a life cycle savings model, the savings ratio (aggregate personal saving divided by disposable income) is explained by per-capita disposable income, the percentage rate of change in per-capita disposable income, and two demographic variables: the percentage of population less than 15 years old and the percentage of the population over 75 years old.

However, we are interested here in the relationship between the two demographic variables (percent of population under 15, percent of population over 75) and the three financial variables (personal savings, per-capita disposal income, growth rate of dpi). The first stage of any such analysis would

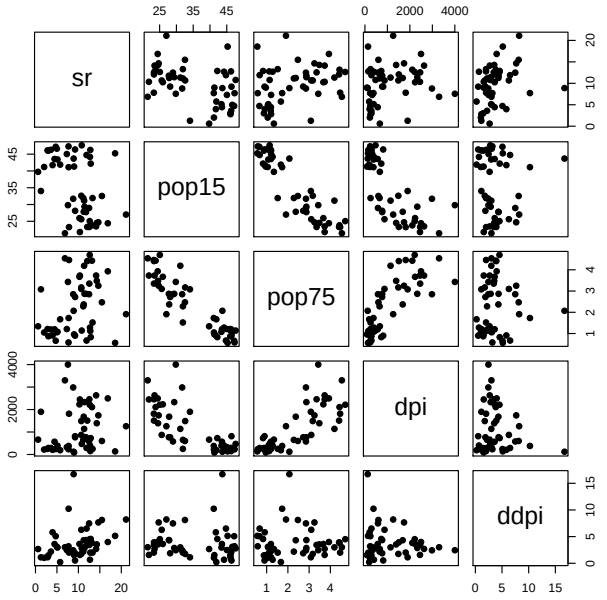


Figure 10.1: Pairwise scatterplots of Life Cycle Savings data

be a visual inspection.

```
pairs(LifeCycleSavings, pch = 16)
```

And it is worth examining the correlation matrix:

```
> cor(LifeCycleSavings)
               sr       pop15       pop75        dpi         ddpi
sr     1.0000000 -0.45553809  0.31652112  0.2203589  0.30478716
pop15 -0.4555381  1.00000000 -0.90847871 -0.7561881 -0.04782569
pop75  0.3165211 -0.90847871  1.00000000  0.7869995  0.02532138
dpi    0.2203589 -0.75618810  0.78699951  1.0000000 -0.12948552
ddpi   0.3047872 -0.04782569  0.02532138 -0.1294855  1.00000000
```

It appears that the **X** variables are correlated. This is less so for **Y** variables, and even less so for **X**,**Y** inter-correlations.

You need to be sure that the variables are *scaled* before carrying out a canonical correlation analysis.

```
LifeCycleSavingsS <- scale(LifeCycleSavings)
pop <- LifeCycleSavingsS[, 2:3] ## The X matrix
oec <- LifeCycleSavingsS[, -(2:3)] ## the Y matrix
```

Having created an **X** matrix and a **Y** matrix, we now want to find linear combinations of **X** which have maximum correlation with **Y**.

```
> cancor(pop, oec)
$cor
[1] 0.8247966 0.3652762

$xcoef
             [,1]       [,2]
pop15 -0.08338007 -0.3314944
pop75  0.06279282 -0.3360027

$ycoef
           [,1]        [,2]         [,3]
sr   0.03795363  0.14955310 -0.023106040
dpi  0.12954600 -0.07518943  0.004502216
ddpi 0.01196908 -0.03520728  0.148898175

$xcenter
        pop15         pop75
-4.662937e-16  2.753353e-16

$ycenter
          sr          dpi         ddpi
1.421085e-16 6.661338e-17 4.440892e-16
```

This indicates one canonical correlate with a correlation of 0.8247966 between $z_{\boldsymbol{X}1}$ and $z_{\boldsymbol{Y}1}$

$$z_{\boldsymbol{X}1} = -0.08338007x_{pop15} + 0.06279282x_{pop75} \tag{10.1}$$

$$z_{\boldsymbol{Y}1} = 0.03795363y_{sr} + 0.12954600y_{dpi} + 0.01196908y_{ddpi} \tag{10.2}$$

If we extract the coefficients as vectors (this time we have created `LCS.cancor` as an object; also we have used `as.numeric(...)` to extract the coefficients in a form suitable for matrix multiplication).

```
> LCS.cancor <- cancor(pop, oec)
> ycoef <- as.numeric(LCS.cancor$ycoef[,1])
> xcoef <- as.numeric(LCS.cancor$xcoef[,1])
> v1 <-  oec %*% ycoef ## remember oec and pop are scaled
> u1 <-  pop %*% xcoef
> plot(v1, u1)
> identify(v1, u1, row.names(LifeCycleSavings))
```

10.3.1 Interpreting the canonical variables

There is some "controversy" about the best way of interpreting the canonical variables. You have two possibilities:

- Interpret the coefficients in a similar way to that used in principal components (problems with collinear variables)
- Calculate the correlation between the canonical and the original variables (doesn't tell you anything about joint contributions)

10.3.2 Hypothesis testing

As with Principal Components, a certain amount of hypothesis testing is possible. The distributional properties of canonical variables is far wilder than principal components - none of the recommended books discuss it. However, the tests can be described. For example, if we wanted to test whether there was any relationship between our two sets of variables:

$$H_0; \boldsymbol{\Sigma}_{12} = \mathbf{0}$$

The Likelihood ratio test leads us to:

$$\Lambda^{\frac{2}{n}} = |\boldsymbol{I} - \boldsymbol{S_{22}^{-1}} \boldsymbol{S_{21}} \boldsymbol{S_{11}^{-1}} \boldsymbol{S_{12}}| = \prod_{i=1}^{k} (1 - r_i^2) \sim \Lambda_{Wilks}(p, n-1-q, q)$$

Using Bartlett's approximation this can yield a chi^2 test:

$$-\left(n - \frac{1}{2}(p+q+3)\right) \log \prod_{i=1}^{k} (1 - r_i^2) \sim \chi^2_{pq}$$

As we've seen before, perhaps we are more interested in finding out how many canonical correlations we need to keep in our analysis. Bartlett also proposed a statistic only s canonical correlations are non-zero:

$$-\left(n - \frac{1}{2}(p+q+3)\right) \log \prod_{i=s+1}^{k} (1 - r_i^2) \sim \chi^2_{(p-s)(q-s)}$$

Chapter 11

Factor analysis

11.1 Role of factor analysis

Most of the development of factor analysis has taken place outside the statistical community, most often in terms of Psychometrics which may partly reflect its origins in the study of intelligence. The earliest cited reference is Spearman (1904). Factor Analysis and Principal Components Analysis are often confused with each other, just to be really awkward there is one method of performing Factor Analysis called Principal Component Extraction. The two methods should never be confused. Principal Components seeks orthogonal projections of the data according the variance maximisation with the hope of achieving some dimension reduction. Factor analysis is all about studying the covariance (or correlation) and is based on a statistical model. We hope to describe the covariance relationships between many variables in terms of a few underlying, unobservable random quantities called factors. If there is a group of highly correlated variables, which in turn are uncorrelated with other variables, perhaps these represent realisations of some underlying phenomena that is responsible for for the observed correlations. It is an attempt to approximate the covarariance matrix $\mathbf{\Sigma}$. It is not highly regarded by many statisticians, but it is used by many others. There are currently many variations on a theme (such as Structural Equation Modelling) which are also very common in many applied literatures.

Whilst we can consider one model for factor analysis, there are two very different fitting methods, neither of which is entirely satisfactory. Having found a solution there are a large number of possible rotations of the solution each of which aim to give the most interpretable solution. In other words, don't be surprised if different computer programs give different "Factor Analysis" solutions for the same data.

Factor Analysis is normally carried out with a view to reification: the investigator usually has a

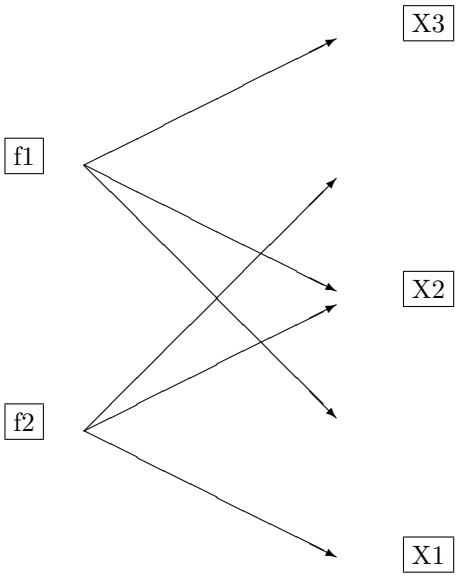


Figure 11.1: Factor analysis, dependence between three variables represented by two latent variables - is this sensible

conceptual model of some underlying entity which cannot be measured directly. These latent, or hidden, variables are the factors in factor analysis. The aim of factor analysis is that each of the p observed variables can be represented by means of $q < p$ mutually uncorrelated common factors. This will leave some uncorrelated residual specific to each of the observed variables, the uniqueness, which is not correlated with any of the remaining $p-1$ variables [1]. It is possible to rotate the q axes of common factors to new orthogonal or obligue axes to make the factor solution fit with existing theoretical ideas regarding the model.

11.2 The factor analysis model

The orthogonal model underlying Factor Analysis can be described as follows:

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\Gamma}\boldsymbol{\phi} + \boldsymbol{\zeta}$$

Where $\boldsymbol{x}$ is an $1 \times p$ random vector. $\boldsymbol{\mu}$ represents a vector of unknown constants (mean values), $\boldsymbol{\Gamma}$ is an unknown $p \times q$ matrix of constants referred to as the *loadings*. $\boldsymbol{\phi}$ is a $q \times 1$ unobserved random vector referred to as the *scores* assumed to have mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma}_{\phi}$, it is commonly assumed that $\boldsymbol{\Sigma}_{\phi} = \boldsymbol{I}$. $\boldsymbol{\zeta}$ is $1 \times p$ unobserved random error vector having mean $\mathbf{0}$ and by assumption a diagonal covariance $\boldsymbol{\psi}$ referred to as the *uniqueness* or *specific variance*.

[1] Note that the diagonal of a correlation matrix is 1. This statement implies that only part of this 1 is due to the $q < p$ latent variables - this part is known as the communality.

With these assumptions, $cov(\boldsymbol{\phi}, \boldsymbol{\zeta}) = 0$, if $\boldsymbol{\Sigma}_\phi = \boldsymbol{I}$ then $cov(\boldsymbol{x}, \boldsymbol{\phi}) = \boldsymbol{\Gamma}$. It is worth emphasising that unlike many multivariate techniques covered here, factor analysis is a statistical model for our observations, with the following distributional form:

$$\boldsymbol{x} \sim Normal(\boldsymbol{\mu}, \boldsymbol{\Gamma}\boldsymbol{\Gamma}^T + \boldsymbol{\psi})$$

It may be slightly clearer to consider the way a vector of observations $\boldsymbol{x} = x_1, \ldots, x_p$ are modelled in factor analysis:

$$\begin{aligned} x_1 &= \mu_1 + \sum_{k=1}^{q} \gamma_{1k}\phi_k + \zeta_1 \\ x_2 &= \mu_2 + \sum_{k=1}^{q} \gamma_{2k}\phi_k + \zeta_2 \\ &\vdots \\ x_p &= \mu_p + \sum_{k=1}^{q} \gamma_{pk}\phi_k + \zeta_p \end{aligned}$$

Note that under the terms of this model:

$$var(x_j) = \gamma_{j1}^2 + \gamma_{j2}^2 + \ldots + \gamma_{jq}^2 + var(\zeta_j) \tag{11.1}$$

One potential problem with this model should be immediately obvious, there can be rather more parameters than data. For example, note that the covariance matrix $\boldsymbol{\Sigma}$ has $p(p+1)/2$ parameters, the factor model $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T + \boldsymbol{\psi})$ has $qp - q(q-1)/2 + p$ parameters. One issue arises whereby a factor analsis model must be constrained in order to ensure identifiability. Clearly, $p(p+1)/2 \geq qp - q(q-1)/2 + p$, or:

$$q \leq \frac{2p + 1 - \sqrt{8p-1}}{2} \tag{11.2}$$

This gives some maximum values of q for given values of p:

p	max q
1	0
2	0
3	1
4	1
5	2
6	3
7	3
8	4
9	5
10	6

Where $q < p$, the right hand side of 11.1 indicates how much of $var(x_j$ is explained by the model, a concept referred to as the communality. Consideration of the order of the model leads on to a point we will consider later, degrees of freedom after fitting a q factor model:

$$df = \frac{p(p+1)}{2} - qp + \frac{q(q-1)}{2} - p = \frac{(p-q)^2 - (d+m)}{2} \tag{11.3}$$

11.2.1 Centred and standardised data

In practice it is often much simpler to centre the data, so that we model:

$$x_j - \mu_j = \sum_{k=1}^{q} \gamma_k \phi_k + \zeta_j; j = 1, \ldots, p \tag{11.4}$$

or even to standardise the variables so that in effect we are modelling the correlation matrix rather than the covariance matrix.

$$\frac{x_j - \mu_j}{\sigma_{jj}} = \sum_{k=1}^{q} \gamma_k \phi_k + \zeta_j; j = 1, \ldots, p \tag{11.5}$$

Regardless of the data matrix used, factor analysis is essentially a model for $\mathbf{\Sigma}$, the covariance matrix of $\boldsymbol{x}$,

$$\mathbf{\Sigma} = \mathbf{\Gamma}\mathbf{\Gamma}^T + \boldsymbol{\psi}$$

11.2.2 Factor indeterminacy

We now consider another problem with factor analysis. It is a very indeterminate model, specifically it is unchanged if we replace $\boldsymbol{\Gamma}$ by $\boldsymbol{K\Gamma}$ for any orthogonal matrix $\boldsymbol{K}$. However, this can be turned to our advantage, with sensible choice of a suitable orthogonal matrix $\boldsymbol{K}$ we can achieve a rotation that may yield a more interpretable answer. Factor analysis therefore requires an additional stage, having fitted the model we may wish to consider rotation of the coefficients.

11.2.3 Strategy for factor analysis

To fit the model, we therefore need to:

- Estimate the number of common factors q.
- Estimate the factor loadings $\boldsymbol{\Gamma}$
- Estimate the specific variances $\boldsymbol{\psi}^2$
- On occasion, estimate the factor scores $\boldsymbol{\phi}$

We will now consider fitting methods for factor analysis. It will be obvious that the preferred method in R is the maximum likelihood method, but we will first consider methods based around principal components to reinforce some ideas about the model.

11.3 Principal component extraction

We have already used the spectral decomposition to obtain one possible factoring of the covariance matrix $\boldsymbol{\Sigma}$.

$$\boldsymbol{\Sigma} = \boldsymbol{E\Lambda E}^T$$

which can be expanded:

$$\begin{aligned}\boldsymbol{\Sigma} &= \lambda_1 \boldsymbol{e}_1 \boldsymbol{e}_1^T + \lambda_2 \boldsymbol{e}_2 \boldsymbol{e}_2^T + \ldots \lambda_p \boldsymbol{e}_p \boldsymbol{e}_p^T \\ &= \left(\sqrt{\lambda_1}\boldsymbol{e}_1, \sqrt{\lambda_2}\boldsymbol{e}_2, \ldots, \sqrt{\lambda_p}\boldsymbol{e}_p\right) \begin{pmatrix} \sqrt{\lambda_1}\boldsymbol{e}_1 \\ \sqrt{\lambda_2}\boldsymbol{e}_2 \\ \vdots \\ \sqrt{\lambda_p}\boldsymbol{e}_p \end{pmatrix}\end{aligned}$$

Of course, in practice we don't know $\boldsymbol{\Sigma}$ and we use S (or we standardise the variables and use R - it should be remembered that this is rather a big decision when working with principal components). Referring back to our data, it should be remembered that the spectral decomposition yields linear principal components as follows:

$$\begin{aligned} z_1 &= e_{11}x_1 + e_{12}x_2 + \ldots + e_{1p}x_p; var(z_1) = \lambda_1 \\ z_2 &= e_{21}x_2 + e_{22}x_2 + \ldots + e_{2p}x_p; var(z_2) = \lambda_2 \\ &\vdots \\ z_p &= e_{p1}x_1 + e_{p2}x_1 + \ldots + e_{1p}x_p; var(z_p) = \lambda_p \end{aligned}$$

which in matrix notation this can be expressed as:

$$\boldsymbol{Z} = \boldsymbol{E}\boldsymbol{X} \tag{11.6}$$

where $\boldsymbol{Z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_p \end{pmatrix}$, $\boldsymbol{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix}$ and $\boldsymbol{E} = \begin{pmatrix} e_{11} & e_{12} & \ldots & e_{1p} \\ e_{21} & e_{22} & \ldots & e_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ e_{p1} & e_{p2} & \ldots & e_{pp} \end{pmatrix}$.

Multiplying both sides of 11.6 by $\boldsymbol{E}^{-1}$gives:

$$\boldsymbol{E}^{-1}\boldsymbol{Z} = \boldsymbol{X} \tag{11.7}$$

We know orthogonal matrices generally that $\boldsymbol{E}^{-1} = \boldsymbol{E}^T$ so we can invert the transformation by using

$$\boldsymbol{X} = \boldsymbol{E}^T\boldsymbol{Z} \tag{11.8}$$

which can be expanded as:

$$\begin{aligned} x_1 &= e_{11}z_1 + e_{21}z_2 + \ldots + e_p z_p \\ x_2 &= e_{12}z_2 + e_{22}z_2 + \ldots + e_p z_p \\ &\vdots \\ x_p &= e_{1p}z_1 + e_{2p}z_2 + \ldots + e_{pp} z_p \end{aligned}$$

which we could express as;

$$\begin{aligned} x_1 &= (e_{11}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{21}\sqrt{\lambda_2})\frac{z_2}{\sqrt{\lambda_2}} + \ldots + (e_{p1}\sqrt{\lambda_p})\frac{z_p}{\sqrt{\lambda_p}} \\ x_2 &= (e_{12}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{12}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + \ldots + (e_{p2}\sqrt{\lambda_p})\frac{z_p}{\sqrt{\lambda_p}} \\ &\vdots \\ x_p &= (e_{1p}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{2p}\sqrt{\lambda_2})\frac{z_2}{\sqrt{\lambda_2}} + \ldots + (e_{pp}\sqrt{\lambda_p})\frac{z_p}{\sqrt{\lambda_p}} \end{aligned}$$

and if we set $\gamma_{jk} = (e_{jk}\sqrt{\lambda_j})$ and $\phi_j = z_j/\sqrt{\lambda_j}$ we have a clear link with the factor analysis model given in equation 11.2. If we try writing this in matrix terminology, our loadings matrix $\mathbf{\Gamma}$ is the $p \times p$ matrix where the jth column is given by $\sqrt{\lambda_j}\boldsymbol{e}_j$ we now have:

$$\boldsymbol{S} = \mathbf{\Gamma}\mathbf{\Gamma}^T$$

which is getting us part of the way to our factor analysis model. Before going any further we will reinforce this procedure by considering how to obtain these values from within R. Note that under the principal component solution, the estimated loadings do not alter as the number of factors is increased or decreased. We are going to load the economic data, and carry out a decomposition of the correlation matrix $\boldsymbol{R}$.

```
> econ <- read.csv("econ.csv", row.names = 1)
> econ.cor <- cor(econ)
> econ.ev <- eigen(econ.cor)
> loadings <- matrix(0,9,9)
> for (i in 1:9){
> loadings[,i] <- sqrt(econ.ev$values[i]) * econ.ev$vectors[,i]
> }
> econ.cor - loadings %*% t(loadings) ## should equal zero
```

Clearly we don't actually want to use a decomposition with with $q = p$ variables. As might be rather obvious bearing in mind our earlier use of principal components, we wish to partition $\mathbf{\Lambda}$ into $\mathbf{\Lambda}_1 = \lambda_1, \lambda_2, \ldots, \lambda_q$ and $\mathbf{\Lambda_2} = \lambda_{q+1}, \ldots, \lambda_p$ with the corresponding eigenvectors. As a consequence, we reduce the size of our $\mathbf{\Gamma}$ matrix, i.e. to neglect the contribution of $\lambda_{q+1}\boldsymbol{e}_{q+1}\boldsymbol{e}_{q+1}^T + \ldots \lambda_p \boldsymbol{e}_p \boldsymbol{e}_p^T$. So when considering our model for the data, we wish to partition our factors as follows:

$$\begin{aligned} x_1 &= e_{11}z_1 + e_{21}z_2 + \ldots + e_{q1}z_q + e_{q+1,1}z_{q+1} + \ldots + e_{p1}z_p \\ x_2 &= e_{12}z_2 + e_{22}z_2 + \ldots + e_{q2}z_q + e_{q+1,2}z_{q+1} + \ldots + e_{p2}z_p \\ &\vdots \\ x_p &= e_{1p}z_1 + e_{2p}z_2 + \ldots + e_{qp}z_q + e_{q+1,p}z_{q+1} + \ldots + e_{pp}z_p \end{aligned}$$

and if we set $e_{q+1,j}z_{q+1} + \ldots + e_{pj}z_p = \zeta_j; j = 1, \ldots, p$ we can rewrite this as:

$$\begin{aligned} x_1 &= e_{11}z_1 + e_{21}z_2 + \ldots + e_{q1}z_q + \zeta_1 \\ x_2 &= e_{12}z_1 + e_{22}z_2 + \ldots + e_{q2}z_q + \zeta_2 \\ &\vdots \\ x_p &= e_{1p}z_1 + e_{2p}z_1 + \ldots + e_{qp}z_q + \zeta_p \end{aligned}$$

As earlier, we can expressed this as:

$$\begin{aligned} x_1 &= (e_{11}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{21}\sqrt{\lambda_2})\frac{z_2}{\sqrt{\lambda_2}} + \ldots + (e_{q1}\sqrt{\lambda_q})\frac{z_q}{\sqrt{\lambda_q}} + \zeta_1 \\ x_2 &= (e_{12}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{12}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + \ldots + (e_{q2}\sqrt{\lambda_q})\frac{z_q}{\sqrt{\lambda_q}} + \zeta_2 \\ &\vdots \\ x_p &= (e_{1p}\sqrt{\lambda_1})\frac{z_1}{\sqrt{\lambda_1}} + (e_{2p}\sqrt{\lambda_2})\frac{z_2}{\sqrt{\lambda_2}} + \ldots + (e_{qp}\sqrt{\lambda_q})\frac{z_q}{\sqrt{\lambda_q}} + \zeta_p \end{aligned}$$

where $\gamma_{jk} = (e_{jk}\sqrt{\lambda_j})$ and $\phi_i = z_i/\sqrt{\lambda_i}$ as before, notice as stated at the outset that $var(\boldsymbol{\zeta}) = \boldsymbol{\psi}$. If we consider this in terms of the decomposition of the covariance matrix we have:

$$\mathbf{\Sigma} = \left(\sqrt{\lambda_1}\boldsymbol{e}_1, \sqrt{\lambda_2}\boldsymbol{e}_2, \ldots, \sqrt{\lambda_q}\boldsymbol{e}_q\right) \begin{pmatrix} \sqrt{\lambda_1}\boldsymbol{e}_1 \\ \sqrt{\lambda_2}\boldsymbol{e}_2 \\ \vdots \\ \sqrt{\lambda_q}\boldsymbol{e}_q \end{pmatrix} + \begin{bmatrix} \psi_1 & 0 & \ldots & 0 \\ 0 & \psi_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \psi_p \end{bmatrix} \tag{11.9}$$

Where now $\psi_j = var(\zeta_j) = \sigma_{jj} - \sum_{k=1}^{q} \gamma_{jk}^2$ for $k = 1, 2, \ldots, q$.

Estimates of the specific variances are given by diagonal elements of the matrix $\hat{\mathbf{\Sigma}} - \hat{\mathbf{\Gamma}}\hat{\mathbf{\Gamma}}^T$, i.e:

$$\hat{\boldsymbol{\psi}} = \begin{bmatrix} \psi_1 & 0 & \ldots & 0 \\ 0 & \psi_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \psi_p \end{bmatrix} \text{with} \psi_j = \sigma_{jj} - \sum_{k=1}^{q} \gamma_{jk}^2 \tag{11.10}$$

So, when using the principal component solution of $\hat{\mathbf{\Sigma}}$, it is specified in terms of eigenvalue-eigenvector pairs $(\hat{\lambda}_1, \hat{\boldsymbol{e}}_1)$, $(\hat{\lambda}_2, \hat{\boldsymbol{e}}_2)$, ..., $(\hat{\lambda}_p, \hat{\boldsymbol{e}}_p)$, where $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \ldots \geq \hat{\lambda}_p$. If we wish to find a $q < p$ solution of common factors, then the estimated factor loadings are given by:

$$\hat{\mathbf{\Gamma}} = \left(\sqrt{\lambda_1}\boldsymbol{e}_1, \sqrt{\lambda_2}\boldsymbol{e}_2, \ldots, \sqrt{\lambda_q}\boldsymbol{e}_q\right)$$

As with the factor analysis model given earlier, the factors $\boldsymbol{\phi}$ have identity covariance matrix

$$var(\boldsymbol{\phi}) = var\left(\sqrt{\Lambda_1}\mathbf{\Gamma}_1^T(\boldsymbol{x} - \boldsymbol{\mu})\right) = \boldsymbol{I}_q,$$

and are uncorrelated with the residuals:

$$cov(\boldsymbol{\phi}, \boldsymbol{\zeta}) = cov\left(\sqrt{\Lambda_1}\mathbf{\Gamma}_1^T(\boldsymbol{x} - \boldsymbol{\mu}), \mathbf{\Gamma}_2\mathbf{\Gamma}_2^T(\boldsymbol{x} - \boldsymbol{\mu})\right) = \sqrt{\mathbf{\Lambda_1}}\mathbf{\Gamma}_1^T\mathbf{\Sigma}\mathbf{\Gamma}_2\mathbf{\Gamma}_2^T = 0$$

However, one major objection to this principal component "solution" is that it can also be seen that each ζ_i contains the same z_i so they are not mutually unrelated. Hence the latent variables obtained using the principal component method do not explain all the correlation structure in our data $\boldsymbol{X}$. The covariance matrix for the errors is now:

$$var(\boldsymbol{\zeta}) = \mathbf{\Gamma}_2\mathbf{\Lambda}_2\mathbf{\Gamma}_2^T$$

This additional step can be carried out fairly easily in R. We only need to discard the unwanted components and estimate the uniquenesses:

```
> loadings4 <- matrix(0,9,4)
> for (i in 1:4){
> loadings4[,i] <- sqrt(econ.ev$values[i]) * econ.ev$vectors[,i]
> }
> LLt <- loadings4 %*% t(loadings4)
> unique <- diag(econ.cor - LLt)
> error <- econ.cor - (LLt + unique)
```

and so `loadings4` gives us the matrix of loadings, `unique` gives us an estimate of the uniquenesses. It should be noted that the loadings are unaltered as the number of factors q is changed. It may be noted that the diagonal elements of $\hat{\boldsymbol{\Sigma}}$ are given by the diagonal elements of $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T + \boldsymbol{\psi}$, but this is not true of the off-diagonal elements. There are error terms associated with our decomposition of the covariance matrix, these can be easily found from `error`. Clearly these are values we wish to see minimised.

We will consider interpretation of factor structure in more detail later. However, for now it may be of interest to examine the four factor solution. It would appear that the first factor represents some kind of contrast between agriculture and other industries (with the exception of finance and mining).

```
\loadings4
                               [,1]        [,2]        [,3]        [,4]
Agriculture               -0.978199871 -0.07760625  0.05173168 -0.02899271
Mining                    -0.002342834 -0.90214224 -0.21179672 -0.06592893
Manufacture                0.645370678 -0.52159027 -0.15703856  0.34982446
PowerSupplies              0.476333161 -0.37897538 -0.58769654 -0.39731951
Construction               0.608061420 -0.07694001  0.15838634  0.66387307
ServiceIndustries          0.707975893  0.51045159 -0.12126845  0.05137022
Finance                    0.138717720  0.66237521 -0.61559512  0.05147600
SocialAndPersonalServices  0.723602099  0.32374238  0.32749903 -0.40851903
TransportAndCommunications 0.684640120 -0.29451591  0.39342807 -0.31637790
```

11.3.1 Diagnostics for the factor model

We can define a residual matrix as:

$$\boldsymbol{\epsilon} = \boldsymbol{S} - \left(\boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{\psi}\right) \tag{11.11}$$

By construction, the diagonal elements of this residual matrix will be zero. A decision to retain a particular q factor model could be made depending on the size of the off-diagonal elements. Rather conveniently, there is an inequality which gives us:

$$\left[\boldsymbol{\epsilon} = \hat{\boldsymbol{\Sigma}} - \left(\boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{\psi}\right)\right] \leq \hat{\lambda}_{q+1}^2 + \cdots + \hat{\lambda}_p^2 \tag{11.12}$$

So it is possible to check the acceptability of fit in terms of a small sum of squares of neglected eigenvalues.

In a similar manner to that used in principal components, it is possible to use the eigenvalues to indicate the proportion of variance explained by any given factor. So instead of examining discarded components we could examine those we intend to retain. Bearing in mind that $trace(\boldsymbol{\Sigma} = \sigma_{11} + \sigma_{22} + \ldots + \sigma_{pp}$, we know that the amount of variation explained by the first factor $\gamma_{11}^2 + \gamma_{21}^2 + \ldots + \gamma_{p1}^2 = (\sqrt{\lambda_1}\boldsymbol{e}_1)^T(\sqrt{\lambda_1}\boldsymbol{e}_1) = \lambda^1$.

So we know that the j-th factor explains the following proportion of total sample variance:

$$\frac{\lambda_j}{trace(\boldsymbol{S})} \tag{11.13}$$

which reduces to $\frac{\lambda_j}{p}$ when using standardised variables (the correlation matrix).

It is actually in the context of factor analysis that the Kaiser criterion was developed. This is implemented by default in a number of computer programs, basically we retain factors which are explaining more than the average amount of variance; if we are decomposing the correlation matrix we retain all factors where the corresponding eigenvalues are greater than one. We can consider the number of components to retain from our earlier eigen analysis. The following R output gives the eigenvalues, and the proportion and cumulating proportion explained by each possible factor.

```
> econ.ev$values
[1]  3.482820 2.132332 1.098373 0.9984261 0.5450933
[6] 0.3836385 0.2224905 0.1367327 0.0000930

> econ.ev$values / 0.09
[1] 38.698003578 23.692581759 12.204146983 11.093622860  6.056592340
[6]  4.262650255  2.472116648  1.519252072  0.001033506

> cumsum(econ.ev$values/0.09)
[1]  38.69800  62.39059  74.59473  85.68836  91.74495  96.00760  98.47971
[8]  99.99897 100.00000
```

Considering the eigenvalues first, using the Kaiser criterion would lead us to select three components, but it should be noted that the fourth component is only just below 1 (0.998) giving perhaps some warning as to the arbitrariness of this device. There are 9 variables, so we divide by 9 (and multiply by 100 to express the proportions as a percentage). Cumulative values are also given. We require five components to explain over 90% of the variation. Remember that according to formula 11.2 this

is the largest value of q that can be contemplated with nine manifest variables.

Communalities

Another important concept are the communalities. In the case of standardised variables, these indicate the proportion of variance of a manifest variable explained by its relevant factor structure. These are simply estimated as:

$$\xi_{jk}^2 = \gamma_{j1}^2 + \gamma_{j2}^2 + \cdots + \gamma_{jq}^2 \tag{11.14}$$

Terminology can now be supplied for the decomposition of the variance of $\boldsymbol{x}$ given earlier in 11.1 to reflect the reduced dimensionality.

$$var(x_j) = \underbrace{\gamma_{j1}^2 + \gamma_{j2}^2 + \ldots + \gamma_{jq}^2}_{communality\ of\ x_j} + \underbrace{\psi_i}_{specificity\ of\ x_j} \tag{11.15}$$

For standardised variables, $var(x_j) = 1$, therefore: $\gamma_{i1}^2 + \gamma_{i2}^2 + \ldots + \gamma_{i1}^q \leq 1$ and $-1 \leq \gamma_{jk} \leq 1$.

These are fairly simply extracted from our matrix of loadings by squaring all entries and summing by row:

```
> row.names(loadings4) <- row.names(econ.cor)
> apply(loadings4^2, 1, sum)
               Agriculture                     Mining
                 0.9664145                 0.8630706
               Manufacture              PowerSupplies
                 0.8355980                 0.8737656
              Construction          ServiceIndustries
                 0.8414721                 0.7791356
                   Finance  SocialAndPersonalServices
                 0.8395906                 0.9025525
TransportAndCommunications
                 0.8103523
```

These appear to be reasonably high for most variables which would suggest a plausible fit for the factor model.

11.3.2 Principal Factor solution

We might have been worried about the way our model above doesn little to account for the off-diagonal elements of $\hat{\boldsymbol{\Sigma}}$. Principal factoring (which seems to have rather fewer advocates) considering decomposing a reduced matrix. We know that the diagonal elements of our covariance matrix are given by $\sigma_{jj} = \xi_j^2 + \psi_j$, so having determined the number q of common factors needed, we can decompose the reduced covariance matrix. If we obtain some initial estimates of $\boldsymbol{\psi}$, we can re-estimate the remaining part of the decomposition $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T$.

$$\sigma_{jj} = \xi_j^2 + \psi_j \tag{11.16}$$

If we had some initial estimate of $\boldsymbol{\psi}$, $\widetilde{\boldsymbol{\psi}}$ say, we could obtained a "reduced" covariance matrix

$$\boldsymbol{S} = \begin{pmatrix} \widetilde{\xi}_1^2 & s_{12} & \cdots & s_{1p} \\ s_{21} & \widetilde{\xi}_2^2 & \cdots & s_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1} & s_{p2} & \cdots & \widetilde{\xi}_p^2 \end{pmatrix}$$

and carry out an eigendecomposition of this matrix, updating our estimates of the uniqueness and repeat until convergence.

So all we need is an initial estimate of $\widetilde{\boldsymbol{\psi}}$. Many programs conduct a multiple regression of each manifest variable on each other, and use $s_{jj}r_j^2$. We then conduct a principal component analysis on $\boldsymbol{S} - \boldsymbol{\psi}$to find $\boldsymbol{\Gamma}$. $\boldsymbol{\psi}$ can then be recalculated as the diagonal of $\boldsymbol{S} - \boldsymbol{\Gamma}\boldsymbol{\Gamma}^T$ and we extract a further set of principal components. These latter steps are repeated until convergence, which can be slow if it happens at all. As we are working with the correlation matrix, it's easy enough to find these intial values:

```
> r2s <- vector("numeric", 9)
>
> for (i in 1:9){
+ y <- econ[,i]
+ x <- econ[,-i]
+ mod <- lm(y~as.matrix(x))
+ r2s[i] <- summary(mod)$r.squared
+ }
>
>
> unique <- diag(1-r2s)
> diag(unique)
[1] 0.0001429627 0.0420230140 0.0006887118 0.1325518542 0.0138610514
[6] 0.0016432160 0.0043879128 0.0007569780 0.0161276459
```

And all that is now required is to repeatedly implement the loop stage. This is presented as a function in `mvmmisc.R`, but it is worth pasting through this manually to see how the procedure works.

```
> new <- econ.cor - unique
> new.ev <- eigen(new)
>
> loadings4pf <- matrix(0,9,4)
> for (i in 1:4){
+ loadings4pf[,i] <- sqrt(new.ev$values[i]) * new.ev$vectors[,i]
+ }
>
> LLt <- loadings4pf %*% t(loadings4pf)
>
> unique.f <- econ.cor - LLt
> diag(unique) <- diag(unique.f)
>
> diag(unique)
[1] 0.02890147 0.14965321 0.15824274 0.25101144 0.14818053 0.21896192 0.15269320
[8] 0.09280381 0.20001567
>
> loadings4pf
             [,1]        [,2]        [,3]         [,4]
 [1,] -0.980556270 -0.06951113  0.06899117 -0.004043332
 [2,] -0.007596438 -0.88584019 -0.20248258  0.156770695
 [3,]  0.644457570 -0.53242421 -0.27602489 -0.258391989
 [4,]  0.453749164 -0.35005068 -0.40926760  0.503055476
 [5,]  0.607710009 -0.08927793 -0.03546047 -0.687953501
 [6,]  0.711408766  0.50862889 -0.12606387 -0.018444548
 [7,]  0.140377243  0.67201322 -0.59955488  0.128581528
 [8,]  0.727082131  0.31778092  0.43171651  0.301966740
 [9,]  0.681111481 -0.30230726  0.45690884  0.189515460
```

It should be noted that in addition to slow (or no) convergence, different results will be obtained depending on whether correlation or covariance matrix is used. However, this approach does not require any distributional assumptions so may be of some use of multivariate normality cannot be claimed, even by refuge to the central limit theorem. Harmon (1967) does indicate further fundamental differences between the principal component and this solution.

Little more needs to be said about this method of factor analysis, we now turn our attention to a more promising approach, maximum likelihood.

11.4 Maximum likelihood solutions

Obvious conclusions might be drawn by noting that R only offers this method of fitting factor analysis models, see the helpfile for the relevant function `?factanal` as well as Venables and B.D.Ripley

(2002). It should be noted from the outset that this method is invariant to changes in scale, a proof given in Seber (1984). In other words, it doesn't matter whether the correlation or the covariance matrix are used, or indeed whether any other scale changes are applied. There are a number of other advantages associated with maximum likelihood fitting, but the problem of Heywood cases still remains, whereby some of the unique variances are estimated with a negative value.

We also need to impose an additional assumption over and above the factor analysis assumptions set out earlier, namely that the following matrix:

$$\boldsymbol{\Gamma}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\Gamma} \tag{11.17}$$

must be diagonal to enable model fitting. Having fitted the model, as we will find out later we are free to rotate the solution.

If the maximum likelihood method is so superior, the obvious question arises as to either of the principal component based methods have remained in use for so long. There is in fact a long and far from trouble free history in terms of trying to develop a maximum likelihood solution for factor analysis, details of an earlier approach to maximum likelihood fitting are given in Morrison (1976). In any case, we well assume that our data follows a multivariate normal distribution, which will have the following likelihood:

$$L(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-\frac{np}{2}}|\boldsymbol{\Sigma}|^{-\frac{n}{2}}e^{-\frac{1}{2}tr\left(\boldsymbol{\Sigma}^{-1}(\sum_{i=1}^{n}(\boldsymbol{x}_i-\bar{\boldsymbol{x}})(\boldsymbol{x}_i-\bar{\boldsymbol{x}})^T+n(\bar{\boldsymbol{x}}-\boldsymbol{\mu})(\bar{\boldsymbol{x}}-\boldsymbol{\mu})^T)\right)} \tag{11.18}$$

we wish to solve this in terms of our factor analysis model and therefore need to find an expression for the likelihood of $L((\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Gamma},\boldsymbol{\psi})$.

$\boldsymbol{\mu}$ is a nuisance parameter for our purposes here, we can either get rid of it by using the estimate $\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{x}}$ and hence use the profile likelihood to find $\hat{\boldsymbol{\Gamma}}$ and $\hat{\boldsymbol{\psi}}$, or we can factorise the likelihood as $L(\boldsymbol{S};\bar{\boldsymbol{x}},\boldsymbol{\Sigma})L(\bar{\boldsymbol{x}};\boldsymbol{\mu},\boldsymbol{\Sigma})$. In this latter case, $\boldsymbol{barx}$ and $\boldsymbol{S}$ are the joint sufficient statistics for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ respectively, for the purposes of factor analysis we only require the first part of the factorised likelihood which can be estimated by conditional maximum likelihood. Note that as $\boldsymbol{barx}$ and $\boldsymbol{S}$ are independent this is also the marginal likelihood.

Taking logs of 11.18, and collecting constant terms into c_1 and c_2 we can say that we wish to maximise:

$$\ln L = c_1 - c_2\left(\ln|\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T+\boldsymbol{\psi}| + trace(\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T+\boldsymbol{\psi})^{-1}\boldsymbol{S}\right) \tag{11.19}$$

By taking this likelihood, along with the diagonality contraints indicated in 11.17 all we need is a procedure for estimation.

An intial estimate of $\widetilde{\psi}$ has to be made as before, Lawley and Maxwell (1971) give maximum likelihood solutions for the uniquenesses. Joreskog (1967) noted that for fixed $\psi > 0$, the likelihood equations require:

$$\hat{\mathbf{\Gamma}} = \sqrt{\psi}\boldsymbol{E}_1\sqrt{(\mathbf{\Lambda}_1 - \boldsymbol{I})} \tag{11.20}$$

where $\mathbf{\Lambda}_1$ contains the q largest eigenvalues of $\sqrt{\psi}\boldsymbol{S}\sqrt{\psi}$, and $\boldsymbol{E}_1$ the corresponding eigenvectors. This is used to estimate $\hat{\mathbf{\Gamma}}$ given a value of $\hat{\psi}$. Now, the log likeihood is maximised with respect to $\hat{\psi}$ given an estimate of $\hat{\mathbf{\Gamma}}$.

As stated, this method is implemented in R, and therefore it is quite easy to try to fit a model to our economics data:

```
> econ <- read.csv("econ.csv", row.names = 1)
> econ.fact <- factanal(econ, factors = 4, rotation = "none")
```

We can consider the residual matrix for our maximum likelihood solution

```
> loadml <- loadings(econ.fact)
> class(loadml) <- "matrix"
> uniqueml <- econ.fact$uniquenesses
> resid <- econ.cor - ( loadml%*% t(loadml) + diag(uniqueml) )
> resid
```

It will be seen that these are considerably smaller than those residuals obtained from the principal component method used earlier. One gain from using the maximum likelihood method is that classical multivariate work provide a test for the adequacy of model fit. If take our null hypothesis as belief that our factor analysis model is an adequate representation of the covariance matrix we will test the following:

$$\begin{aligned} H_0 &: \mathbf{\Sigma} = \mathbf{\Gamma}\mathbf{\Gamma}^T + \psi \\ H_1 &: \mathbf{\Sigma} \; is \; any \; other \; positive \; definite \; matrix \end{aligned}$$

This (eventually) yields a likelihood ratio statistic:

$$-2\ln\Lambda = -2\ln\left(\frac{|\hat{\mathbf{\Sigma}}|}{|\boldsymbol{S}|}\right) + n\left(tr(\hat{\mathbf{\Sigma}}^{-1}\boldsymbol{S}) - p\right) \tag{11.21}$$

with $\frac{1}{2}\left((p-q)^2-p-q\right)$ degrees of freedom.

It can be shown (not here) that $tr(\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{S})-p=0$ at the maximum likelihood so this term can be removed and we can consider that

$$-2\ln\Lambda = n\ln\left(\frac{|\hat{\boldsymbol{\Sigma}}|}{|\boldsymbol{S}|}\right) \tag{11.22}$$

All that remains is to add a correction suggested by Bartlett (1951, 1954). We need to replace n with something slightly more elaborate, the exact formula chosen varies amongst many multivariate tests, in the current R function the correction applied is:

$$n-1-\frac{2p+5}{6}-\frac{2q}{3}$$

Hence we are going to test:

$$n-1-\frac{2p+5}{6}-\frac{2q}{3}\ln\left(\frac{|\hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Gamma}}^T+\hat{\boldsymbol{\psi}}|}{|\boldsymbol{S}|}\right) > \chi^2_{((p-q)^2-p-q)/2,\alpha} \tag{11.23}$$

The idea might be to start with q small (anticipating the rejection of H_0), and increase q until H_0 is no longer rejected. As with all such tests, there are many reasons for rejecting H_0, not all of these may concern us. In addition, Johnson and Wichern (1998) suggest that if n is large and q is small relative to p, it will tend to reject H_0 even though $\hat{\boldsymbol{\Sigma}}$ is close to $\boldsymbol{S}$. So the situation can arise whereby we can claim "statistical significance" for the inclusion of additional factors in our model, but they actually add little to the model. This tends to reinforces the exploratory aspects of multiviariate analysis (for some sense of exploratory).

We can extract the communalities from our model as easily as before:

```
> apply(loadml^2, 1, sum)
               Agriculture                     Mining
                 0.9954167                  0.5937541
               Manufacture              PowerSupplies
                 0.9950263                  0.9950022
              Construction          ServiceIndustries
                 0.4852833                  0.8360147
                   Finance  SocialAndPersonalServices
                 0.4786655                  0.9950786
TransportAndCommunications
                 0.4676025
```

The R output has already given us information on the proportion of variance explained by each of the factors:

```
                Factor1 Factor2 Factor3 Factor4
SS loadings       3.270   1.519   1.189   0.864
Proportion Var    0.363   0.169   0.132   0.096
Cumulative Var    0.363   0.532   0.664   0.760
```

suggesting that 76% of variation is explained by our four factors (under the maximum likelihood solution). We reckoned on 10% points more for the principal component solution. This would be expected due the variance maximising properties of principal components generally (whether used appropriately or for factor analysis).

It is now important to turn our attention to rotation. The maximum likelihood solution is constrained by the diagonality constraint, and it is particularly important here that rotations are considered.

11.5 Rotation

It was stated earlier that one of the potential disadvantages of factor analysis was a certain rotational indeterminancy, indeed in the maximum likelihood fitting method it is necessary to add a constraint specifically deal with this. We are now going to consider one of the benefits of rotation; to yield a more interpretable factor structure. In short, we seek a rotation:

$$\hat{\mathbf{\Gamma}}^{(R)} = \hat{\mathbf{\Gamma}}\boldsymbol{T} \tag{11.24}$$

such that we obtain easy-to-interpret factor loadings. One definition of "easy" is that where possible some components would be large, others small. The most obvious way to do this is actually to carry out the exercise by eye, and to rotate the axes around the origin so that some factor loadings become small. It is also easy to suggest a two dimensional rotation matrix:

$$\boldsymbol{T} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$$

for rotation angle $\phi; -\pi \leq \phi \leq \phi$. All we need to do is find a suitable value for ϕ. This becomes slightly more difficult in every sense where $q > 2$, indeed it is possible to carry out the whole procedure by eye with pencil and paper (do you remember what they are).

For orthogonal rotations, two objective criteria are most commonly used to determine the optimal rotation: the Varimax procedure (Kaiser, 1958) and the Quartimax procedure Neuhaus and Wrigley

(1954). The former is currently available within R and will be considered here, as usual it is worth checking the definitive entry in `?varimax`. This looks for a rotation which maximises the objective V:

$$V = \frac{1}{p^2}\sum_{k=1}^{q}\left(p\sum_{j=1}^{p}\left[\frac{\gamma_{jk}^2}{\xi_j^2}\right]^4 - \left[\sum_{j=1}^{p}\left[\frac{\gamma_{jk}^2}{\xi_j^2}\right]\right]^2\right) \tag{11.25}$$

where $\xi_i^2 = \sum_{k=1}^{q}\gamma_{jk}^2$ is the communality for each of the j variables as before.

Earlier we called `factanal()` with the argument `rotation = "none"`, hence the default is to carry out a rotation. It is also possible to obtain a promax rotation. However, it is useful for our purposes to carry out the rotations directly on the loadings matrices we have generated earlier, the following call:

```
> varimax(loadings4)
```

will supply a varimax rotation of our four principal component factors.

In many books dealing with topic it is conventional to consider this subject by visually rotating the axis, leaving the loadings in the same position. However, inverting this procedure we can very simply plot the rotated and unrotated loadings as follows:

```
> plot(loadings4[,c(1:2)], pch = as.character(c(1:9)),
    xlab = expression(paste(gamma,"1")), ylab = expression(paste(gamma,"2")),
    main = "First and second loadings",
    xlim = c(-1,1), ylim = c(-1,1))
> points(varimax(loadings4)$loadings[,c(1:2)],
    pch = letters[c(1:9)], col = "red")
> abline(h = 0)
> abline(v = 0)
```

where the numbers 1-9 represent the unrotated loadings for variables 1 to 9, and the letters a-i represent the rotated loadings for variables 1 to 9 on the first two factors. This is depicted in figure 11.2.

Although is is more difficult to see what is going on with $q = 4$, we can see for example that the eighth variable (Social and Personal Services) has in increased loading in terms of γ_{18}, and a much decreased loading in terms of the second factor (γ_{28} is virtually zero. Thus we may feel that we have achieved some simplification of our factor structure.

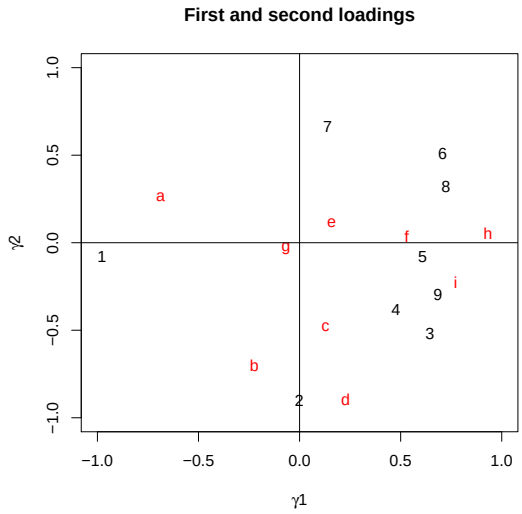


Figure 11.2: Plot overlaying the co-ordinates of factor loadings 1 and 2 before and after rotation optimised by the varimax criterion

11.6 Factor scoring

Finally, there are occasions where we may wish to estimate values for $\boldsymbol{\phi}_i$ for a given individual i. These values are referred to as the scores, the process of estimating them, which has to be carried out after $\boldsymbol{\Gamma}$ and $\boldsymbol{\psi}$ have been estimated is therefore referred to as scoring.

Two methods are available in R for scoring, Thomson and Bartlett's. The default is that no scoring takes place (it requires a data matrix). By including `scores = "Bartlett"`) or `scores = "regression"` these estimates are obtained.

Bartlett (1937, 1938) propsed a method based upon weighted least squares.

Once we have estimates

$$\begin{aligned} x_1 - \bar{x}_1 &= \sum_{k=1}^{q} \hat{\gamma}_{1k}\phi_1 + \zeta_1 \\ x_2 - \bar{x}_2 &= \sum_{k=1}^{q} \hat{\gamma}_{2k}\phi_2 + \zeta_2 \\ &\vdots \\ x_p - \bar{x}_p &= \sum_{k=1}^{q} \hat{\gamma}_{pk}\phi_p + \zeta_p \end{aligned}$$

we need to estimate ϕ_j for $j = 1, \ldots, q$, however as $var(\zeta_j) = \psi_j$ are not equal he argued that weighted least squares was the most appropriate technique.

The weighted least squares estimates thus obtained are:

$$\hat{\boldsymbol{\phi}}_i = (\boldsymbol{\Gamma}^T \boldsymbol{\Psi}^{-1} \boldsymbol{\Gamma}) \boldsymbol{\Gamma}^T \boldsymbol{\Psi} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \tag{11.26}$$

Thomson (1951) is based on assuming that both $\boldsymbol{\phi}$ and $\boldsymbol{\zeta}$ are multivariate normal, thus a concatenation of the manifest ($\boldsymbol{x}$) and latent ($\boldsymbol{\phi}$) variables $\boldsymbol{y}^T = (\boldsymbol{\phi}^T, \boldsymbol{x}^T)$ will also be normal with dispersion matrix:

$$var(\boldsymbol{y}) = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{\Gamma}^T \\ \boldsymbol{\Gamma} & \boldsymbol{\Gamma}\boldsymbol{\Gamma}^T + \boldsymbol{\psi} \end{pmatrix}$$

The mean of $\boldsymbol{\phi}$ is zero by definition, therefore:

$$E(\boldsymbol{z}|\boldsymbol{x}_0) = \boldsymbol{\Gamma}^T (\boldsymbol{\Gamma}) \boldsymbol{\Gamma}^T + \boldsymbol{\Psi})^{-1} (\boldsymbol{x}_o - \boldsymbol{\mu})$$

which gives the estimate for the scores as:

$$\boldsymbol{z} = \hat{\boldsymbol{\Gamma}}^T (\hat{\boldsymbol{\Gamma}}) \hat{\boldsymbol{\Gamma}}^T + \hat{\boldsymbol{\psi}})^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{mu}}) \tag{11.27}$$

It might be clear that factor scoring takes no account of uncertainty in the estimates of $\hat{\boldsymbol{\Gamma}}$ and $\hat{\boldsymbol{\psi}}$, this is one area where Bayesian methods are coming to the fore (Aitkin and Aitkin, 2005)

Bayesian factor analysis, sparsity priors

Bibliography

Afifi, A. and V. Clark (1990). *Computer-aided Multivariate Analysis* (2nd ed.). New York: Van Nostrand Reinhold.

Aitkin, M. and I. Aitkin (2005). Bayesian inference for factor scores. In A. Maydeu-Olivares and J. J. McArdle (Eds.), *Contemporary Advances in Psychometrics*. New York: Erlbaum.

Anderson, T. (1963). Asymptotic theory for principal component analysis. *Annals of Mathematical Statistics 34*, 122–148.

Anderson, T. (1984). *An Introduction to Multivariate Statistical Analysis* (2 ed.). New York: John Wiley.

Atkinson, A. C., M. Riani, and A. Cerioli (2004). *Exploring multivariate data with the forward search*. New York: Springer Verlag.

Bartlett, M. (1937). The statistical conception of mental factors. *British Journal of Psychology 28*, 97–104.

Bartlett, M. (1938). Methods of estimating mental factors. *Nature 141*, 609–610.

Bartlett, M. (1951). The effect of standardisation on an approximation in factor analysis. *Biometrika 38*, 337–344.

Bartlett, M. (1954). A note on multiplying factors for various chi-squared approximations. *J. R. Statist. Soc. B 16*, 296–298.

Bilodeau, M. and D. Brenner (1999). *Theory of Multivariate Statistics*. New York: Springer.

Bolton, R. J. and W. J. Krzanowski (1999). A characterization of principal components for projection pursuit. *The American Statistician 53*(2), 108–109.

Box, G. and D. Cox (1964). An analysis of transformations. *J. R. Statist. Soc. B 26*, 211–252.

Brandeau, M. and S. Chiu (1988). Parametric facility location in a tree network with a $l_p$norm cost function. *Transportation Science 22*, 59–69.

Bumpus, H. (1898). The elimination of the unfit as illustrated by the introduced sparrow *Passer domesticus*. Biological Lectures, Marine Biology Laboratory, Woods Hole, 11th Lecture, pp.209-226.

Cailliez, F. (1983). The analytical solution of the additive constant problem. *Psychometrika 48*, 343–349.

Cattell, R. (1966). The scree test for the number of factors. *Multivariate Behavioural Research 1*, 245–276.

Chatfield, C. and A. Collins (1980). *Introduction to Multivariate Analysis*. London: Chapman and Hall.

Dice, L. (1945). Measures of the amount of ecological association between species. *Journal of Ecology 26*, 297–302.

Dillon, W. and M. Goldstein (1984). *Multivariate Analysis - Methods and Applications*. New York: Wiley.

Eastment, H. and W. Krzanowski (1982). Cross-validatory choice of the number of components from a principal component analysis. *Technometrics 24*, 73–78.

Eckart, C. and G. Young (1936). The approximation of one matrix by another of lower rank. *Psychometrika 1*, 211–218.

et al., J. F. H. (1998). *Multivariate data analysis* (5th ed.). Upper Saddle River, NJ: Prentice Hall International.

Everitt, B. and G. Dunn (1991). *Applied Multivariate Data Analysis*. London: Edward Arnold.

Everitt, B., S. Landau, and M. Leese (2001). *Cluster Analysis* (4th ed.). London: Arnold.

Fleiss, L. and J. Zubin (1969). On the methods and theory of clustering. *Multivariate Behavioural Research 4*, 235–250.

Flury, B. (1988). *Common principal components and related multivariate models*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. New York: John Wiley & Sons Inc.

Flury, B. (1997). *A First Course in Multivariate Statistics*. New York: Springer.

Flury, B. and H. Riedwyl (1988). *Multivariate statistics : a practical approach*. Chapman and Hall.

Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics 21*, 768–769.

Gabriel, K. (1971). The biplot graphical display of matrices with applications to principal component analysis. *Biometrika 58*, 453–467.

Gentleman, R., B. Ding, S. Dudoit, and J. Ibrahim (2005). Distance measures in DNA microarray data analysis. In R. Gentleman, V. Carey, W. Huber, R. A. Irizarry, and S. Dudoit (Eds.), *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, Chapter 12, pp. 189–208. New York: Springer.

Giri, N. (2003). *Multivariate Statistical Analysis*. New York: Marcel Dekker Inc.

Girshink, M. (1939). On the sampling theory of roots of determinantal equations. *Annals of Mathematical Statistics 10*, 203–224.

Gnanadesikan, R. (1977). *Methods for Statistical Data Analysis of Multivariate Observations*. New York: Wiley.

Gnanadesikan, R. (1997). *Methods for Statistical Data Analysis of Multivariate Observations* (2nd ed.). New York: Wiley.

Gnanadesikan, R., J. Harvey, and J. Kettenring (1993). Mahalanobis metrics for cluster analysis. *Sankhya Series A Special Volume 55*, 494–505.

Gordon, A. (1999). *Classification* (Second Edition ed.). London: Chapman and Hall / CRC.

Gower, J. (1971). A general coefficient of similarity and some of its properties. *Biometrics 27*, 857–871.

Gower, J. (1984). Distance matrices and their euclidean approximation. In E. Diday, M. Jambu, L. Lebart, J. Pages, and R. Tomassone (Eds.), *Data Analysis and Informatics 3*, pp. 3–21. Amsterdam: North-Holland.

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika 53*, 325–328.

Hair, J., R. Anderson, R. Tatham, and W. Black (1995). *Multivariate Data Analysis with Readings* (4th ed.). Englewood Cliffs, NY: Prentice Hall.

Harmon, H. (1967). *Modern Factor Analysis*. Chicago: The University of Chicago Press.

Harris, R. (1985). *A Primer on Multivariate Statistics* (2 ed.). Orlando: Academic Press.

Hartigan, J. A. and M. A. Wong (1979). A k-means clustering algorithm. *Applied Statistics 28*, 100–108.

Healy, M. (2000). *Matrices for Statistics* (2nd ed.). Oxford: Clarendon Press.

Horn, J. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika 30*, 179–185.

Hotelling, H. (1931). The generalization of student's ratio. *Annals of Mathematical Statistics 2*, 360–378.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology 24*, 417–441;498–520.

Jackson, J. (1991). *A User's Guide to Principal Components*. New York: Wiley.

Jardine, N. and R. Sibson (1971). *Mathematical Taxonomy*. Chichester: Wiley.

John, S. (1971). Some optimal multivariate tests. *Biometrika 58*, 123–127.

John, S. (1972). The distribution of a statistic used for testing sphericity of normal distributions. *Biometrika 59*, 169–173.

Johnson, D. (1998). *Applied Multivariate Methods for Data Analysts*. Pacific Grove, CA: Duxbury Press.

Johnson, R. and D. Wichern (1998). *Applied Multivariate Statistical Analysis* (4th ed.). New Jersey: Prentice Hall.

Jolicoeur, P. and J. Mosimann (1960). Size and shape variation in the painted turtle: A principal component analysis. *Growth 24*, 339–354.

Jolliffe, I. (1986). *Principal Components Analysis*. New York: Springer-Verlag.

Jolliffe, I., B. Jones, and B. Morgan (1986). Comparison of cluster analysis of the english personal social services of authorites. *J. R. Statist. Soc. A 149*(3), 253–270.

Joreskog, K. (1967). Some contributions to maximum likelihood factor analysis. *Psychometrika 32*, 443–482.

Kaiser, H. (1958). The varimax criterion for the analytic rotation in factor analysis. *Psychometrika 23*, 187–200.

Karlis, D., G. Saporta, and A. Spinakis (2003). A simple rule for the selection of principal components. *Communications in Statistics: Theory and Methods 32*(3), 643–666.

Kaufman, L. and P. J. Rousseeuw (1989). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley.

Kendall, M. (1975). *Multivariate Analysis*. High Wycombe, England: Charles Griffin and Company Ltd.

Krause, E. (1975). *Taxicab Geometry*. Menlo Park, CA: Addison Wesley.

Kruskal, J. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrica 29*, 1–27.

Krzanowski, W. and F. Marriott (1994a). *Multivariate Analysis: Part 1 Distributions, Ordination and Inference*. Kendals Library of Statistics. London: Edward Arnold.

Krzanowski, W. and F. Marriott (1994b). *Multivariate Analysis: Part 2 Classification, Covariance Structures and Repeated Measurements*. Kendals Library of Statistics. London: Edward Arnold.

Krzanowski, W. J. (2000). *Principles of Multivariate Analysis: A users perspective (Revised edition)*. Oxford: Oxford University Press.

Lance, G. and W. Williams (1966). A generalised sorting strategy for computer classifications. *Nature 212*, 218.

Lance, G. and W. Williams (1967). A general theory of classificatory sotring strategies i. hierarchical systems. *Comput.J. 9*, 373–380.

Lance, G. and W. Williams (1979). INVER: a program for the computation of distance measures between attributes of mixed types. *Australian Computer Journal 11*, 27–28.

Larson, R. and G. Sadiq (1983). Facility locations with the manhattan matric in the presence of barriers to travel. *Operations Research 31*, 652–699.

Lawley, D. (1956). Tests of significance of the latent roots of covariance and correlation matrices. *Biometrika 43*, 128–136.

Lawley, D. (1963). On testing a set of correlation coefficients for equality. *Annals of Mathematical Statistics 34*, 149–151.

Lawley, D. and A. Maxwell (1971). *Factor Analysis as a Statistical Method* (2nd ed.). Butterworths.

Ledoit, O. and M. Wolf (2002). Some hypothesis tests for the covariance matrix when the dimension is large compared to the sample size. *Ann. Statist. 30*, 1081–1102.

Lingoes, J. (1971). Some boundary conditions for a monotone analysis of symmetric matrices. *Psychometrika 36*, 195–203.

Lloyd, S. (1957). Least squares quantization in pcm. technical note, bell laboratories. *Published in 1982 in IEEE Transactions on Information Theory 28*, 128–137.

Lubischew, A. (1962). On the use of discriminant functions in taxonomy. *Biometrics*.

Macnaughton-Smith, P., W. Williams, M. Dale, and L. Lockett (1964). Dissimilarity analysis: A new technique of hierarchical subdivision. *Nature 202*, 1034–1035.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, pp. 281–297. Berkeley, CA: University of California Press.

Mahalanobis, P. (1930). On tests and measures of group divergence. *Journal and Proceedings of the Asiatic Society of Bengal 26*, 541–588.

Manly, B. (1994). *Multivariate Statistical Methods: A Primer* (2nd ed.). London: Chapman and Hall.

Mardia, K. (1978). Some properties of classical multidimensional scalling. *Communications in Statistics - Theory and Methods A7*, 1233–1241.

Mardia, K., J. Kent, and J. Bibby (1979). *Multivariate Analysis*. London: Academic Press.

Maronna, R. and P. Jacovkis (1974). Multivariate clustering procedures with variable metrics. *Biometrics 30*, 499–505.

Mauchly, J. (1940). Significance tests for sphericity of a normal n-variate distribution. *Annals of Mathematical Statistics 11*, 204–209.

McLachlan, G. J. and D. Peel (2000). *Finite Mixture Models*. New York: J. Wiley and Sons.

Morrison, D. (1976). *Multivariate Statistical Methods* (2nd ed.). New York: McGraw Hill.

Morrison, D. (2005). *Multivariate Statistical Methods* (4th ed.). Belmont, CA: Brooks / Cole.

Muirhead, R. (New York). *Aspects of Multivariate Statistical Theory*. Wiley.

Neuhaus, J. and C. Wrigley (1954). The quartimax method: an analytical approach to orthogonal simple structure. *Br.J.Statist.Psychol. 7*, 81–91.

Pearson, K. (1901). On lines and planes of closest fit to a system of points in space. *Philosophical Magazine 2*, 557–572.

Press, S. (1982). *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference*. Krieger Publishing Company.

Rencher, A. (2002). *Methods of Multivariate Analysis* (2nd ed.). New York: Wiley.

Richardson, M. (1938). Multidimensional psychophysics. *Psychol.Bull. 35*, 659–660.

Rogers, D. and T. Tanimoto (1960). A computer program for classifying plants. *Science 132*, 1115–1118.

Rothkopf, E. (1957). A measure of stimulus similarity and errors in some paried-associate learning tasks. *Journal of Experimental Psychology 53*, 94–101.

Sammon, J. (1969). A non-linear mapping for data structure analysis. *IEEE Trans. Comput. C-18*, 401–409.

Schott, J. R. (Ed.) (1997). *Matrix Analysis for Statistics*. New York: Wiley.

Schott, J. R. (2005). Miscellanea: Testing for complete independence in high dimensions. *Biometrika 92*(4), 951–956.

Schott, J. R. (2006). A high-dimensional test for the equality of the smallest eigenvalues of a covariance matrix. *Journal of Multivariate Analysis 97*, 827–843.

Seber, G. (1984). *Multivariate Observations*. New York: Wiley.

Sneath, P. (1957). The application of computers to taxonomy. *Journal of General Microbiology 17*, 201–226.

Sokal, R. and C. Michener (1958). A statistical method for evaluating systematic relationships. *Univ.Kansas Sci.Bull. 38*, 1409–1438.

Sokal, R. and P. Sneath (1963). *Principles of Numerical Taxonomy*. San Francisco: Freeman.

Spearman, C. (1904). 'general intelligence', objectively determined and measured. *American Journal of Psychology 15*, 201–293.

Srivastava, M. and E. Carter (1983). *An Introduction to Applied Multivariate Statistics*. New York: North-Holland.

Struyf, A., M. Hubert, and P. J. Rousseeuw (1997). Integrating robust clustering techniques in S-PLUS. *Computational Statistics and Data Analysis* (26), 17–37.

Sugiyama, T. and H. Tong (1976). On a statistic useful in dimensionality reduction in multivariate linear stochastic system. *Commun.Statist.Theor.Meth.A 5*, 711–721.

Thomson, G. H. (1951). *The Factorial Analysis of Human Ability*. Longon University Press.

Timm, N. (1975). *Multivariate Analysis with applications in Education and Psychology*. Belmont CA: Wadsworth Publishing Company.

Torgerson, W. (1952). Multi-dimensional scaling: I - theory and method. *Psychometrika 17*, 401–419.

Tukey, J. (1957). On the comparative anatomy of transformations. *Annals of Mathematical Statistics 28*, 602–632.

Venables, W. and B.D.Ripley (2002). *Modern Applied Statistics with S* (4th ed.). New York: Springer.

Ward, J. (1963). Hierarchical grouping to optimize and objective function. *J. Am. Statist. Ass. 58*, 236–244.

Wickens, T. D. (1995). *The Geometry of Multivariate Statistics*. New Jersey: Lawrence Erlbaum Associates.

Wold, S. (1976). Pattern recognition by means of disjoint principal component models. *Pattern Recognition 8*, 127–139.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics 20*, 397–405.

Young, G. and A. Householder (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika 3*, 19–22.