Introduction to R for Multivariate Data Analysis

Fernando Miguez

July 9, 2007

email: miguez@uiuc.edu
office: N-211 Turner Hall
office hours: Wednesday 12pm or by appointment

1 Introduction

This material is intended as an introduction to the study of multivariate statistics and no previous knowledge of the subject or software is assumed. My main objective is that you become familiar with the software and by the end of the class you should have an intuitive understanding of the subject. I hope that you also enjoy the learning process as much as I do. I believe that for this purpose R and GGobi will be excellent resources. R is free, open source, software for data analysis, graphics and statistics. Although GGobi can be used independently of R, I encourage you to use GGobi as an extension of R. GGobi is a free software for data visualization (www.ggobi.org). This software is under constant development and it still has occasional problems. However, it has excellent capabilities for learning from big datasets, especially multivariate data. I believe that the best way to learn how to use it is by using it because it is quite user friendly. A good way of learning GGobi is by reading the book offered on-line and the GGobi manual, which is also available from the webpage. It is worth saying that GGobi can be used from R directly by a very simple call once the `rggobi` package is installed. In the class we will also show examples in SAS which is the leading commercial software for statistics and data management.

This manual is a work in progress. I encourage you to read it and provide me with feedback. Let me know which sections are unclear, where should I expand the content, which sections should I remove, etc. If you can contribute in any way it will be greatly appreciated and we will all benefit in the process. Also, if you want to directly contribute, you can send me texts and figures and I will be happy to include it in this manual.

Let us get to work now and I hope you have fun!

1.1 Installing R

http://www.r-project.org/
Select a mirror and go to `"Download and Install R"`
These are the steps you need to follow to install `R` and GGobi.

```
1) Install R.
2) type in the command line
source("http://www.ggobi.org/download/install.r")
3) In this step make sure you install the GTk environment and you
 basically just choose all the defaults.
4) You install the rggobi package from within R by going to
Packages ... Install packages ... and so on.
```

1.2 First session and Basics

When R is started a prompt `>` indicates that it is ready to receive input. In a question and answer manner we can interrogate in different ways and receive an answer. For example, one of the simplest tasks in R is to enter an arithmetic expression and receive a result.

```
> 5 + 7
[1] 12  # This is the result from R
```

Probably the most important operation in R is the assignment, `<-`. This is used to create objects and to store information, data or functions in these objects. An example:

```
res.1 <- 5 + 7
```

Now we have created a new object, `res.1`, which has stored the result of the arithmetic operation 5 + 7. The name `res.1` stands for "result 1", but virtually any other name could have been used. A note of caution is that there are some conventions for naming objects in R. Names for objects can be built from letters, digits and the period, as long as the first symbol is not a period or a number. R is case sensitive. Additionally, some names should be avoided because they play a special role. Some examples are: `c, q, t, C, D, F, I, T, diff, df, pt`.

1.3 Reading and creating data frames

1.3.1 Creating a data frame

There are many ways of importing and/or creating data in R. We will look at just two of them for simplicity. The following code will prompt a spreadsheet where you can manually input the data and, if you want, rename columns,

```
dat <- data.frame(a=0,b=0,c=0)
fix(dat)
# These numbers should be entered 2 5 4 ; 4 9 5 ; 8 7 9
```

The previous code created an object called `dat` which contains three columns called a, b, and c. This object can now be manipulated as a matrix if all the entries are numeric (see **Matrix operations** below).

1.3.2 Importing data

One way of indicating the directory where the data file is located is by going to `"File"` and then to `"Change dir..."`. Here you can browse to the location where you are importing the data file from. In order to create a data frame we can use the following code for the data set `COOKIE.txt` from the textbook.

```
cook <- read.table("COOKIE.txt",header=T,sep="",na.strings=".")
cook[1:3,]
pairs(cook[,5:9])
```

The second line of code selects the first three rows to be printed (the columns were not specified, so all of them are printed). The `header` option indicates that the first row contains the column names. The `sep` option indicates that entries are separated by a space. Comma separated files can be read in this way by indicating a comma in the sep option or by using the function `read.csv`. You can learn more about this by using help (e.g. `?read.table`, `?read.csv`). The third line of code produces a scatterplot matrix which contains columns 5 through 9 only. This is a good way of starting the exploration of the data.

1.4 Matrix operations

We will create a matrix **A** (R is case sensitive!) using the `as.matrix` function from the data frame `dat`. The numeral, #, is used to add comments to the code.

```
A <- as.matrix(dat)
Ap <- t(A)              # the function t() returns the transpose of A
ApA <- Ap %*% A         # the function %*% applies matrix multiplication
Ainv <- solve(A)        # the function solve can be used to obtain the inverse
library(MASS)           # the function ginv from the package MASS calculates
                        # a Moore-Penrose generalized inverse of A
Aginv <- ginv(A)
Identity <- A %*% Ainv # checking the properties of the inverse
round(Identity)         # use the function round to avoid rounding error
sum(diag(A))            # sums the diagonal elements of A; same as trace
det(A)                  # determinant of A
```

Now we will create two matrices. The first one, **A**, has two rows and two columns. The second one, **B**, has two rows and only one column.

```
A <- matrix(c(9,4,7,2),2,2)
B <- matrix(c(5,3),2,1)
C <- A + 5
D <- B*2
DDD <- A %*% B          # matrix multiplication
DDDD <- A * A           # elementwise multiplication
E <- cbind(B,c(7,8))    # adds a column
F <- rbind(A,c(7,8))    # adds a row
dim(A)                  # dimensions of a matrix
O <- nrow(A)            # number of rows
P <- ncol(A)            # number of columns
rowSums(A)              # sums for each column
Mean <- colSums(A)/O    # mean for each column
```

For additional introductory material and further details about the R language see [7].

1.5 Some important definitions

Some concepts are important for understanding statistics and multivariate statistics in particular. These will be defined informally here:

- *Random Variable*: the technical definition will not be given here but it is important to notice that unlike the common practice with other mathematical variables, a random variable cannot be assigned a value; a random variable does not describe the actual outcome of a particular experiment, but rather describes the possible, as-yet-undetermined outcomes in terms of real numbers. As a convention we normally use X for a random variable and x as a realization of that random variable. So x is an actual number even if it is unknown.

- *Expectation*: the expectation of a continuous random variable is defined as,

$$E(X) = \mu = \int x f(x) dx$$

  You can think of the expectation as a mean. The integration is done over the domain of the function, which in the case of the normal goes from $-\infty$ to ∞.

- *Mean vector*: A mean vector is the elementwise application of the Expectation to a column vector of random variables.

$$E(\mathbf{X}) = \begin{bmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_p) \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix} = \boldsymbol{\mu}$$

- *covariance matrix*:

$$\boldsymbol{\Sigma} = E(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})'$$

or

$$\boldsymbol{\Sigma} = Cov(\mathbf{X}) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix}$$

1.6 Eigenvalues and eigenvectors

In multivariate data analysis many methods use different types of decompositions with the aim of describing, or explaining the data matrix (or, more typically the variance-covariance or correlation matrix). Eigenvalues and eigenvectors play an important role in the decomposition of a matrix. The definition of these terms and the theory can be found in the notes or the textbook. Here we will see the functions in R used to carry out these computations. Try the following,

```
A <- matrix(c(1,1,0,3),2,2)
eigen(A)
dm <- scan()
# Insert 13 -4 2 -4 13 -2 2 -2 10
B <- matrix(dm,3,3)
eigen(B)
C <- edit(data.frame())
# enter these numbers
   16    15   20
   10     6   15
   10     6   10
    8     7    5
   10     6   18
    9     8   15
Cm <- as.matrix(C)        # coerce the data frame into a matrix
mean(Cm)                  # mean vector
var(Cm)                   # variance-covariance matrix
cor(Cm)                   # correlation matrix
```

1.7 Distance

The concept of distance can be defined as a spatial difference between two points. So, in a general sense distance is a difference, and it is a more abstract concept that mere spatial difference. For example, we could be interested in the difference in weight of two items. Some methods in this class will involve distance between points in multidimensional space. Most datasets involve variables that suggest spatial distance, others temporal distance and some might not imply a distance that can be measured quantitatively. These later variables are usually named categorical (or classification) variables.

More formally, if we consider the point $P = (x_1, x_2)$ in the plane, the Euclidean distance, $d(O, P)$, from P to the origin $O = (0, 0)$ is, according to the Pythagorean theorem,

$$d(O, P) = \sqrt{x_1^2 + x_1^2} \tag{1}$$

In general if the point P has p coordinates so that $P = (x_1, x_2, \dots, x_p)$, the Euclidean distance from P to the origin $O = (0, 0, \dots, 0)$ is

$$d(O, P) = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2} \tag{2}$$

All points $(x_1, x_2, \dots, x_p)$ that lie a constant squared distance, such as c^2, from the origin satisfy the equation

$$d^2(O, P) = x_1^2 + x_2^2 + \cdots + x_p^2 = c^2 \tag{3}$$

Because this is the equation of a hypershpere (a circle if $p = 2$), points equidistant from the origin lie on a hypershpere. The Euclidean distance between two arbitrary points P and Q with coordinates $P = (x_1, x_2, \dots, x_p)$ and $Q = (y_1, y_2, \dots, y_p)$ is given by

$$d(P, Q) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_p - y_p)^2} \tag{4}$$

This measure of distance is not enough for most statistical methods. It is necessary, therefore, to find a 'statistical' measure of distance. Later we will see the multivariate normal distribution and interpret the statistical distance in that context.

For further details about this section see [5].

1.8 Univariate graphics and summaries

A good way of exploring data is by using graphics complemented by numerical summaries. For example, we can obtain summaries for columns 5 through 9 from the `COOKIE` data set.

```
summary(cook[,5:9])         # 5 numbers summary
boxplot(cook[,5:9])         # boxplots for 5 variables
plot(density(cook[,5]))     # density plot for variables in column 5
```

At this point it is clear that for a large dataset, this type of univariate exploration is only marginally useful and we should probably look at the data with more powerful graphical approaches. For this, I suggest turning to a different software: GGobi. While GGobi is useful for data exploration, R is preferred for performing the numerical part of multivariate data analysis.

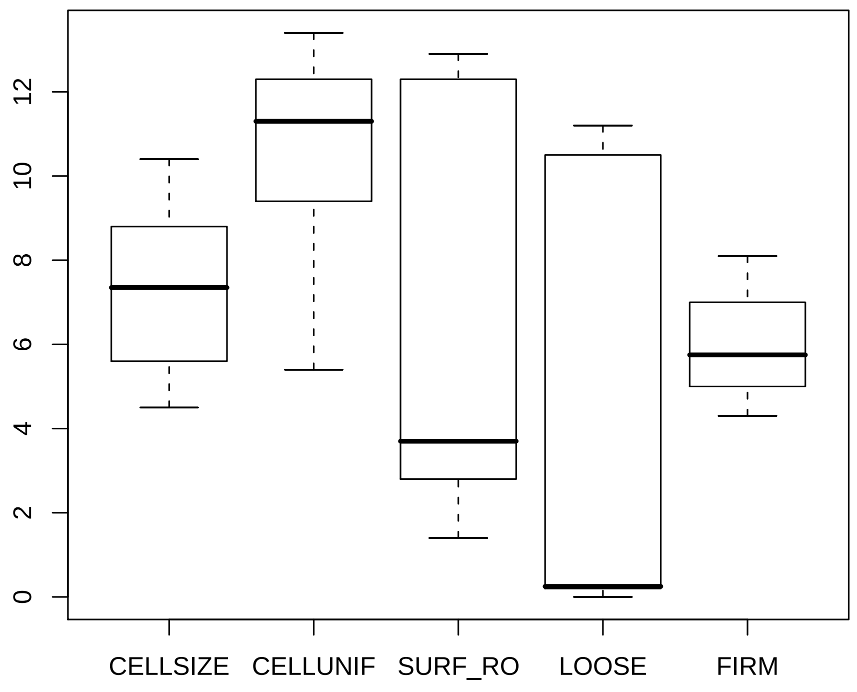


Figure 1: Boxplots for 5 variables in `COOKIE` dataset

2 Multivariate data sets

Typical multivariate data sets can be arranged into a data matrix with rows and columns. The rows indicate 'experimental units', 'subjects' or 'individuals', which will be referred as units from now on. This terminology can be applied to animals, plants, human subjects, places, etc. The columns are typically the variables measured on the units. For most of the multivariate methods treated here we assume that these variables are continuous and it is also often convenient to assume that they behave as random events from a normal distribution. For mathematical purpose, we arrange the information in a matrix, $\mathbf{X}$, of N rows and p columns,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \cdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}$$

where N is the total number of 'units', p is the total number of variables, and x_{ij} denotes the entry of the j^{th} variable on the i^{th} unit. In R you can find example data sets that are arranged in this way. For example,

```
dim(iris)
[1] 150 5
iris[c(1,51,120),]
    Sepal.Length   Sepal.Width  Petal.Length  Petal.Width      Species
1            5.1           3.5           1.4          0.2       setosa
51           7.0           3.2           4.7          1.4   versicolor
120          6.0           2.2           5.0          1.5    virginica
```

We have asked R how many dimensions the `iris` dataset has using the `dim` function. The dataset has 150 rows and 5 columns. The rows are the experimental units, which in this case are plants. The first four columns are the response variables measured on the plants and the last column is the `Species`. The next command requests printing rows 1,51 and 120. Each of these is an example of the three species in the dataset. A nice visual approach to explore this dataset is the `pairs` command, which produces Figure 2. The code needed to produce this plot can be found in R by typing `?pairs`. Investigate!

3 Multivariate Normal Density

Many real world problems fall naturally within the framework of normal theory. The importance of the normal distribution rests on its dual role as both population model for certain natural phenomena and approximate sampling distribution for many statistics [5].

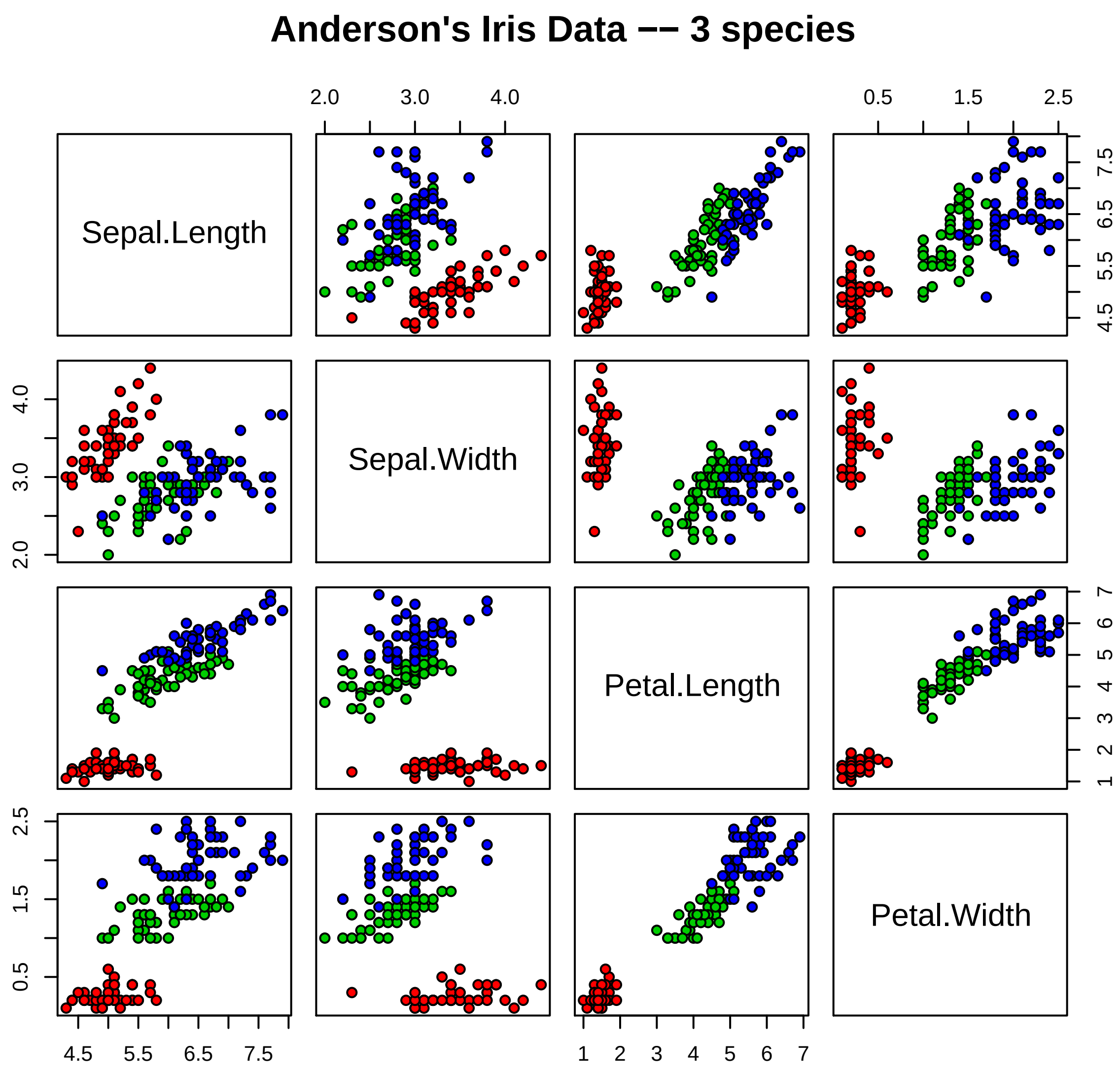


Figure 2: Scatterplot matrix for `iris` dataset

3.1 The multivariate Normal Density and its Properties

The univariate normal is a probability density function which completely determined by two familiar parameters, the mean μ and the variance σ^2,

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp^{-[(x-\mu)/\sigma]^2/2} \quad -\infty < x < \infty$$

The term in the exponent

$$\left(\frac{x-\mu}{\sigma}\right)^2 = (x-\mu)(\sigma^2)^{-1}(x-\mu)$$

of the univariate normal density function measures the square of the distance from x to μ in standard deviation units. This can be generalized for a $p \times 1$ vector $\mathbf{x}$ of observations on several variables as

$$\boldsymbol{(x-\mu)'\Sigma^{-1}(x-\mu)}$$

The $p \times 1$ vector $\boldsymbol{\mu}$ represents the expected value of the random vector $\mathbf{X}$, and the $p \times p$ matrix $\mathbf{\Sigma}$ is the varaince-covariance matrix of $\mathbf{X}$. Given that the symmetric matrix $\mathbf{\Sigma}$ is positive definite the previous expression is the square of the generalized distance from $\mathbf{x}$ to $\boldsymbol{\mu}$. The univariate normal density can be generalized to a $p-$dimensional normal density for the random vector $\mathbf{X}' = [X_1, X_2, \ldots, X_p]$

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

where $-\infty < x_i < \infty, i = 1, 2, \ldots, p$. This $p-$dimensional normal density is denoted by $N_p(\mu, \mathbf{\Sigma})$.

3.2 Interpretation of statistical distance

When $\mathbf{X}$ is distributed as $N_p(\mu, \mathbf{\Sigma})$,

$$\boldsymbol{(x-\mu)'\Sigma^{-1}(x-\mu)}$$

is the squared statistical distance from $\mathbf{X}$ to the population mean vector $\boldsymbol{\mu}$. If one component has much larger variance than another, it will contribute less to the squared distance. Moreover, two highly correlated random variables will contribute less than two variables that nearly uncorrelated. Essentially, the use of the inverse of the correlation matrix, (1) standardaizes all of the variables and (2) eliminates the effects of correlation [5].

3.3 Assessing the Assumption of Normality

It is important to detect moderate or extreme departures from what is expected under multivariate normality. Let us load the `Cocodrile` data into R and investigate multivariate normality. I will select variables `sl` through `wn` to keep the size of the individual scatter plots informative.

```
Croco <- read.csv("Cocodrile.csv", header=T, na.strings=".")
names(Croco)
[1] "id"      "Species" "cl"      "cw"      "sw"      "sl"      "dcl"
[8] "ow"      "iow"     "ol"      "lcr"     "wcr"     "wn"
pairs(Croco[,6:13])
```

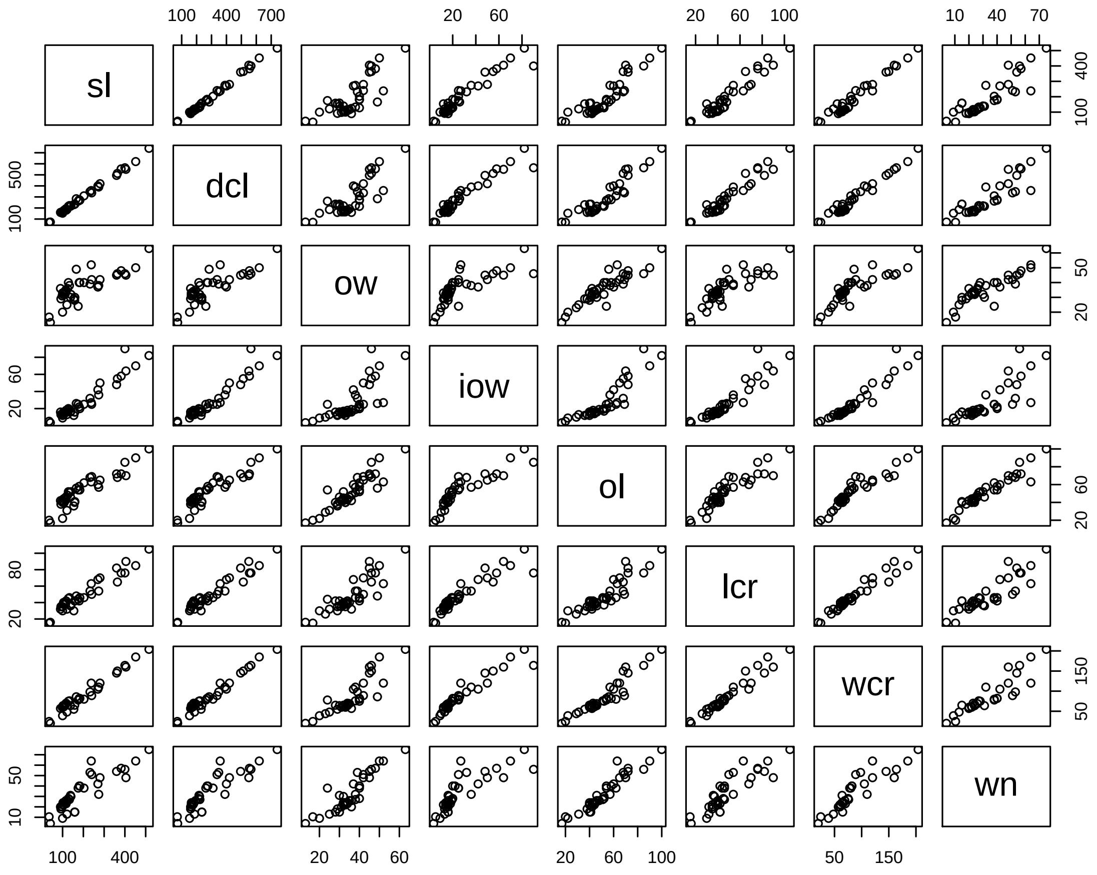


Figure 3: Scatterplot matrix for `Cocodrile` dataset

This data set is probably too clean, but it is a good first example where we see the high degree of correlation among the variables and it seems plausible to assume bivariate normality in each one of the plots and therefore multivariate normality for the data set in general. Let us turn to another example using the `Skull.csv` data set. We can basically follow the same steps as we did for the first dataset. The resulting scatterplot matrix reveals the unusually high correlation between `BT` and `BT1`. `BT1` is a corrected version of `BT` and therefore we should include only one of them in the analysis.

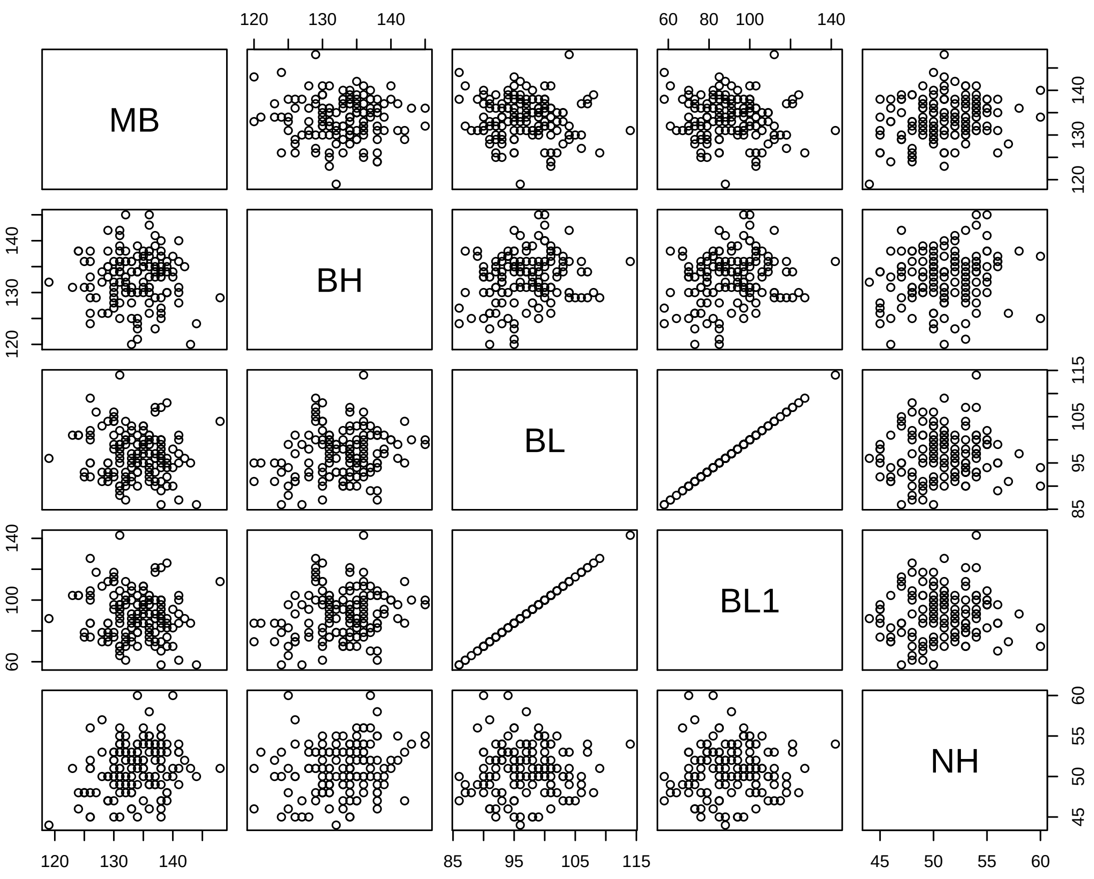


Figure 4: Scatterplot matrix for `Skull` dataset

4 Principal Components Analysis

Principal component analysis (PCA) tries to explain the variance-covariance structure of a set of variables through a *few* linear combinations of these variables [2]. Its general objectives are: data reduction and interpretation. Principal components is often more effective in summarizing the variability in a set of variables when these variables are highly correlated. If in general the correlations are low, PCA will be less useful. Also, PCA is normally an intermediate step in the data analysis since the new variables created (the predictions) can be used in subsequent analysis such as multivariate regression and cluster analysis. It is important to point out that the new variables produced will be uncorrelated.

The basic aim of principal components analysis is to describe the variation in a set of correlated variables, $x_1, x_2, \ldots, x_p$, in terms of a new set of uncorrelated variables $y_1, y_2, y_3, \ldots, y_p$, each of which is a linear combination of the x variables. The new variables are derived in increasing order of "importance" in the sense that y_1 accounts for as much of the variation in the original data amongst all linear combinations of $x_1, x_2, \ldots, x_p$. Then y_2 is chosen to account for as much as possible of the remaining variation, subject to being uncorrelated with y_1 and so on. The new variables defined by this process, $y_1, y_2, \ldots, y_p$ are the principal components.

The general hope of principal components analysis is that the first few components will account for a substantial proportion of the variation in the original variables and can, consequently, be used to provide a convenient lower-dimensional summary of these variables that might prove useful in themselves or as inputs to subsequent analysis.

4.1 Principal Components using only two variables

To illustrate PCA we can first perform this analysis on just two variables. For this purpose we will use the `crabs` dataset.

```
crabs <- read.csv("australian-crabs.csv",header=T)
```

Select variables `"FL" and "RW"`.

```
crabs.FL.RW <- crabs[c("FL","RW")]
```

Scale (or standardize) variables. This is convenient for latter computations.

```
crabs.FL.RW.s <- scale(crabs.FL.RW)
```

Calculate the eigenvalues and eigenvectors.

```
eigen(cor.crabs)
$values
[1] 1.90698762 0.09301238

$vectors
          [,1]       [,2]
[1,] 0.7071068  0.7071068
[2,] 0.7071068 -0.7071068
```

Following we will analyze the APLICAN data in R, which is also analyzed in the class textbook.

```
appli <- read.table("APPLICAN.txt",header=T,sep="")
colnames(appli) <- c("ID","FL","APP","AA","LA","SC","LC","HON",
"SMS","EXP","DRV","AMB","GSP","POT","KJ","SUIT")
```

First I added the column names as this was not supplied in the original dataset.

```
dim(appli)
[1] 48 16
(appli.pc <- princomp(appli[,2:16]))
summary(appli.pc)
```

Compare the output from R with the results in the book. Notice that R does not produce a lot of output (this is true in general and I like it). The "components" are also standard deviations and they are the squared root of the Eigenvalues that you see in the textbook. Let's do a little of R gymnastics to get the same numbers as the book.

```
appli.pc$sdev^2
    Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6
66.1420263 18.1776339 10.5368803  6.1711461  3.8168324  3.5437661
```

I'm showing only the first 6 components here. With the `summary` method we can obtain the Proportion and Cumulative shown on page 104 in the text book. The SCREE plot can be obtain using the `plot` function. The `loadings` function will give you the output on page 105.

This is a good oportunity to use GGobi from R.

```
library(rggobi)
ggobi(appli)
```

5 Factor Analysis

5.1 Factor Analysis Model

Let $\mathbf{x}$ be a p by 1 random vector with mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$. Suppose the interrelationships between the elements of $\mathbf{x}$ can be explained by the *factor model*

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{f} + \boldsymbol{\eta}$$

where $\mathbf{f}$ is a random vector of order k by 1 ($k < p$), with the elements $f_1, \ldots, f_k$, which are called the *common factors*; $\boldsymbol{\Lambda}$ is a p by k matrix of unknown constants, called *factor loadings*; and the elements, $\eta_1, \ldots, \eta_p$ of the p by 1 random vector $\boldsymbol{\eta}$ are

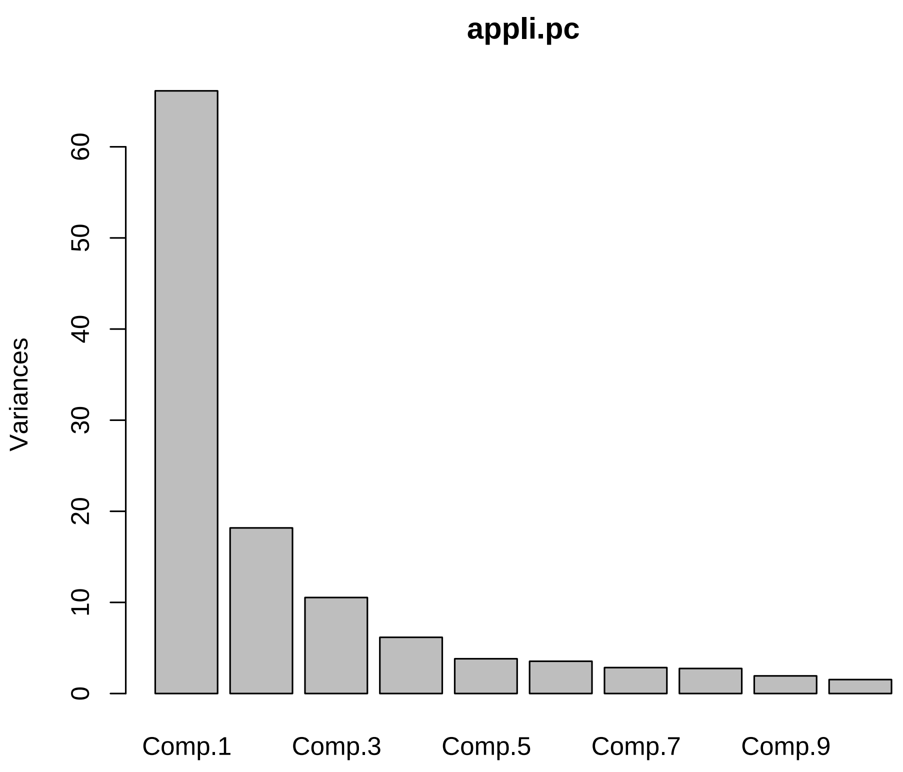


Figure 5: SCREE plot of the eigen values for `APPLICAN` dataset

called the *specific factors*. It is assumed that the vectors $\mathbf{f}$ and $\boldsymbol{\eta}$ are uncorrelated. Notice that this model resembles the classic regression model with the difference that $\boldsymbol{\Lambda}$ is unknown.

The assumptions of this model can be summarized

1. $\mathbf{f} \sim (\mathbf{0}, \mathbf{I})$
2. $\boldsymbol{\eta} \sim (\mathbf{0}, \boldsymbol{\Psi}), \text{where } \boldsymbol{\Psi} = diag(\psi_1, \ldots, \psi_p)$ and
3. $\mathbf{f}$ and $\boldsymbol{\eta}$ are independent.

The assumptions of the model then lead to

$$\boldsymbol{\Sigma} = \mathbf{LL}' + \boldsymbol{\Psi}$$

Under this model the communalities and more generally the factorization of $\boldsymbol{\Sigma}$ remain invariant of an orthogonal transformation. Hence, the factor loadings are determined only up to an orthogonal transformation [4].

5.2 Tests example

Before I introduce an example in R let me say that different software not only produce different results because they might be using different methods but they even produce very different results using apparently the same method. So it should be of no surprise if you compare the results from this analysis with SAS, SPSS, SYSTAT and you see large differences.

This is an example I made up and it is based on tests that were taken by 100 individuals in arithmetic, geometry, algebra, 200m run, 100m swim and 50m run. First we create the object based on the data

```
> test <- read.table("MandPh.txt",header=T,sep="")
> names(test)
[1] "Arith" "Geom"  "Alge"  "m200R" "m100S" "m50R"
```

The correlation matrix is

```
> round(cor(test),2)
      Arith Geom Alge m200R m100S m50R
Arith  1.00 0.88 0.78  0.19  0.24 0.25
Geom   0.88 1.00 0.90  0.16  0.18 0.18
Alge   0.78 0.90 1.00  0.19  0.20 0.20
m200R  0.19 0.16 0.19  1.00  0.90 0.78
m100S  0.24 0.18 0.20  0.90  1.00 0.89
m50R   0.25 0.18 0.20  0.78  0.89 1.00
```

This matrix clearly shows that the 'math' tests are correlated with each other and the 'physical' tests are correlated with each other, but there is a very small correlation between any of the 'math' and the 'physical'.

There are different ways of entering the data for factor analysis. In this case we will just use the object we created in the first step. The only method in R for conducting factor analysis is maximum likelihood and the only rotation available is "varimax".

```
> test.fa1 <- factanal(test, factor=1)
> test.fa1
...
Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 306.91 on 9 degrees of freedom.
The p-value is 8.91e-61
> test.fa2 <- factanal(test, factor=2)
...
Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 5.2 on 4 degrees of freedom.
The p-value is 0.267
```

The fact that we do not reject the null hypothesis with two factors indicates that two factors are needed and this makes sense considering the way we generated the data. We can store the loadings in an object called `Lambda.hat` and the uniqueness in an object called `Psi.hat` and recreate the correlation matrix.

```
> Lambda.hat <- test.fa2$loadings
> Psi.hat <- diag(test.fa2$uni)
> Sigma.hat <- Lambda.hat%*%t(Lambda.hat) + Psi.hat
> Sigma.hat
> round(Sigma.hat,2)
      Arith Geom Alge m200R m100S m50R
Arith  1.00 0.88 0.80  0.21  0.24 0.23
Geom   0.88 1.00 0.90  0.16  0.18 0.18
Alge   0.80 0.90 1.00  0.18  0.20 0.19
m200R  0.21 0.16 0.18  1.00  0.90 0.80
m100S  0.24 0.18 0.20  0.90  1.00 0.89
m50R   0.23 0.18 0.19  0.80  0.89 1.00
```

Notice that in this case `Sigma.hat` is really the fitted correlation matrix and it differs slightly from the original. This difference is the "lack of fit" which will be seen in more detail in the next example.

5.3 Grain example

This examples was taken from a book by Khattree and Naik (1999).

Sinha and Lee (1970) considered a study from agricultural ecology where composite samples of wheat, oats, barley, and rye from various locations in the Canadian prairie were taken. Samples were from commercial and governmental terminal elevators at Thunder Bay (Ontario) during the unloading of railway boxcars. The objective of the study was to determine the interrelationship, if any, between arthropod infestation and grain environment. The three grain environmental variables observed for 165 samples were as follows: grade of sample indicating grain quality (1 highest, 6 lowest) (GRADE), percentage of moisture content in grain (MOIST), and dockage, which measures the presence of weed seed, broken kernels, and other foreign matters (DOCK).

Sometimes all we have is a correlation matrix. In this case we need to enter the data in a special way. The key here is the object (`ecol.cov`) we created as a `list` which contains the covariance (in this case the correlation) matrix. Sometimes we have the information about means and this would be entered as the `center` but in this case they are just zeros, and the number of observations.

Notice that the default in the `factanal` function is to rotate the loadings using the `"varimax"` rotation.

```
ecol <- read.table("ecol_cor.txt",header=T,sep="")
row.names(ecol) <- c("grade","moist","dock","acar","chey","glyc"
,"lars","cryp","psoc")
ecol.cov <- list(cov=ecol.m,center=rep(0,9),n.obs=165)
ecol.FA <- factanal(factors=2, covmat=ecol.cov , n.obs=165)
ecol.FA

Call:
factanal(factors = 2, covmat = ecol.cov, n.obs = 165)
Uniquenesses:
grade moist  dock  acar  chey  glyc  lars  cryp  psoc
0.456 0.423 0.633 0.903 0.693 0.740 0.350 0.951 0.905
Loadings:
      Factor1 Factor2
grade  0.150   0.722
moist  0.556   0.517
dock           0.606
acar   0.291   0.109
chey   0.543   0.109
glyc   0.498   0.112
lars   0.799   0.105
cryp   0.120   0.185
psoc  -0.200  -0.234
               Factor1 Factor2
SS loadings      1.652   1.292
Proportion Var   0.184   0.144
Cumulative Var   0.184   0.327
```

```
Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 36.64 on 19 degrees of freedom.
The p-value is 0.00881
```

We can also produce a plot (see Figure 6) of the rotated factor loadings and verify that they are consistent with the correlations and the biological interpretation.

```
loadings <- ecol.FA$loadings
plot(loadings,type="n",xlim=c(-1,1),ylim=c(-1,1))
abline(h=0,v=0)
text(loadings,labels=row.names(loadings),col="blue")
```

In addition we can get the residual matrix $Res = \hat{P} - \hat{\Lambda}\hat{\Lambda}' + \hat{\Psi}$ and evaluate the lack of fit. Notice that the largest discrepancies are seen for the pair `glyc,cryp` and `glyc,psoc`.

```
Psi.hat <- diag(fecol.FA$uniq)
P <- loadings%*%t(loadings) + Psi.hat
res <- ecol.m - P
round(res,2)
      grade moist  dock  acar  chey  glyc  lars  cryp  psoc
grade  0.00 -0.02  0.00 -0.02  0.03 -0.05  0.01  0.05 -0.04
moist -0.02  0.00  0.02  0.03 -0.04  0.07 -0.01  0.00  0.01
dock   0.00  0.02  0.00 -0.03 -0.01  0.01  0.00 -0.06  0.07
acar  -0.02  0.03 -0.03  0.00  0.01 -0.03 -0.02 -0.04 -0.12
chey   0.03 -0.04 -0.01  0.01  0.00 -0.06  0.03  0.05  0.05
glyc  -0.05  0.07  0.01 -0.03 -0.06  0.00 -0.01 -0.19 -0.18
lars   0.01 -0.01  0.00 -0.02  0.03 -0.01  0.00  0.04  0.05
cryp   0.05  0.00 -0.06 -0.04  0.05 -0.19  0.04  0.00 -0.03
psoc  -0.04  0.01  0.07 -0.12  0.05 -0.18  0.05 -0.03  0.00
```

6 Discriminant Analysis

The problem of separating two or more groups is sometimes called *discrimination* [4, 5, 7]. The literature can be sometimes confusing but this problem is now generally termed *supervised classification* [3]. This is in contrast to *unsupervised classification* which will be covered latter as *cluster analysis*. The term *supervised* means that we have information about the membership of each individuals. For example, we might have a ‘gold standard’ to determine if a plant is infected with a specific pathogen and would like to know if there is additional information that can help us discriminate between these two categories (i.e. *infected* or *not infected*). The classic example of discrimination is with the `iris` example. For this example we have known species of *Iris* and would like to know if measurements made on the flowers can help us discriminate the known three species of *Iris*. I think that in fact Fisher (or Anderson) had some specific things he/they wanted to prove with this dataset, but I’m sure

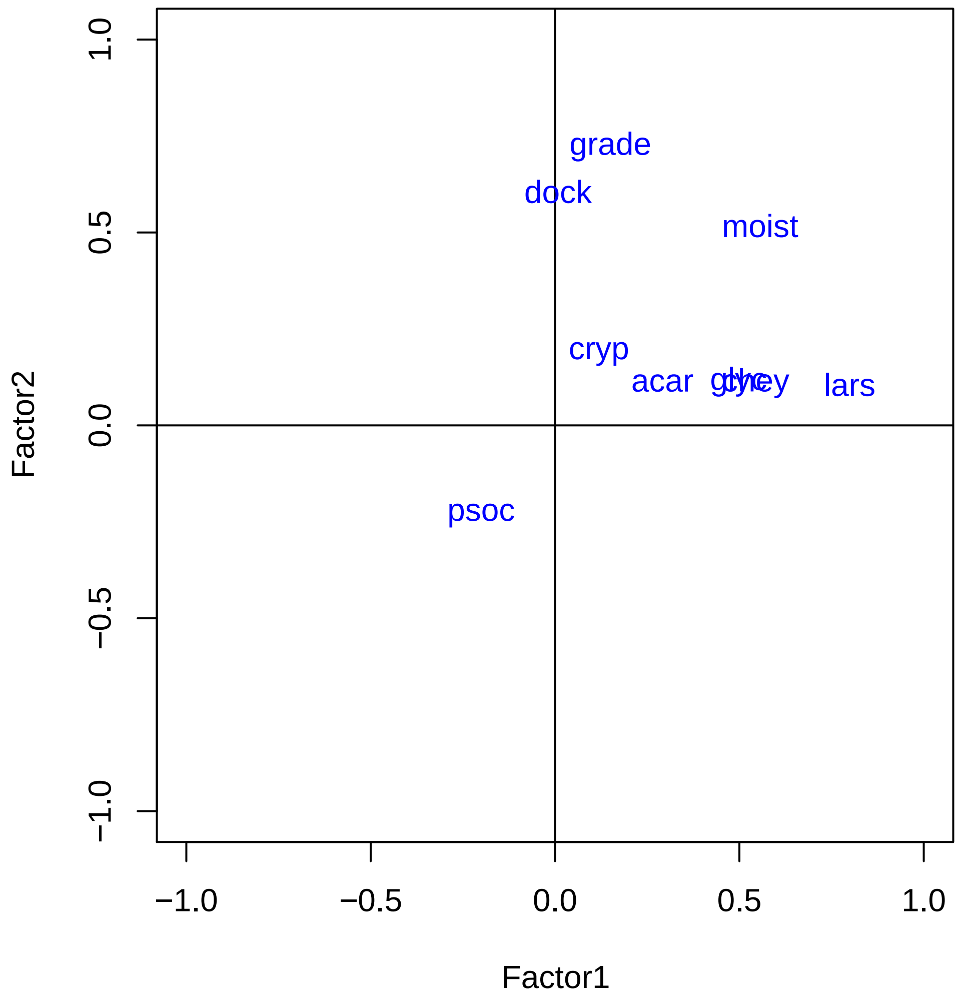


Figure 6: Plot of factor loadings for the grain quality data

you/me can google it and then learn everything about this famous example. An alternative is to do `?iris` in R to find references and such.

There are a couple of issues regarding discrimination. First, we want to make as few mistakes as possible. Making mistakes is, in general, inevitable. A misclassification occurs when we classify a member (say π_1) to group 1 when in reality it should have been assigned to group 2. Second, we might be concerned with the cost of misclassification. Suppose that classifying π_1 object as belonging to π_2 represents a more serious error than classifying a π_2 object as belonging to π_1. Then one should be cautious about making the former assignment.

Next I will show how to analyze the `TURKEY1` dataset which the book analyzes in SAS. There are a few things to notice. The data is not in the format we need for analysis. So we need to do a bit of manipulation prior to the analysis. This is a good exercise to practice object manipulations.

```
> turkey <- read.table("TURKEY1.txt",header=T,sep="",na.strings=".")
> dim(turkey)
[1] 158  19
> turkey2 <- turkey[,c(1,2,3,5,6,7,8,10,11,12,15,19)]
> turkey3 <- turkey2[turkey2$SEX == "MALE",]
> library(MASS)
> tur.nmd <- na.omit(turkey3)
> dim(tur.nmd)
[1] 33 12
> tur.lda <- lda(tur.nmd[,4:12],tur.nmd[,3])
```

Notice that although we started with 158 rows and 19 columns the data we are really analyzing has 33 rows and 12 columns. Missing data is not typically used in this analysis.

In this case we have two groups and we can only have one linear discriminant. A nice plot can be obtained by

```
> plot(tur.lda)
```

The estimated minimum expected cost of misclassification rule for two normal populations is

Allocate $\mathbf{x}_0$ to π_1 if

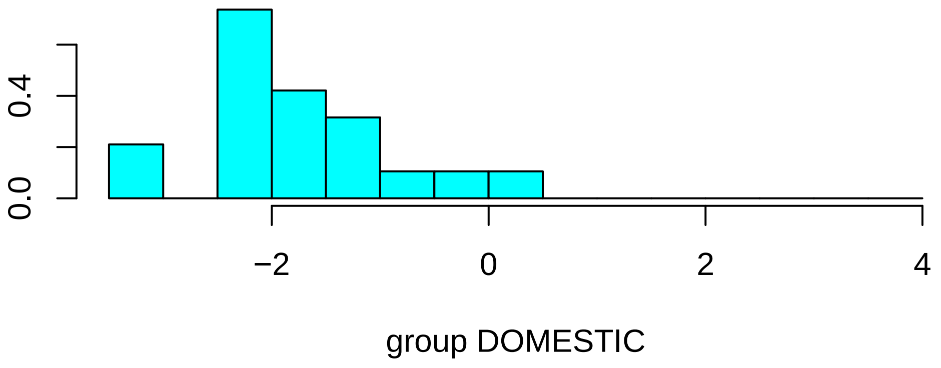
$$(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}\mathbf{x}_0 - \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) \geq ln\left[\left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)\right]$$

Allocate $\mathbf{x}_0$ to π_2 otherwise.

$\mathbf{S}_{pooled}$ is defined as

$$\left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1)}\right]\mathbf{S}_1 + \left[\frac{n_2 - 1}{(n_1 - 1) + (n_2 - 1)}\right]\mathbf{S}_2$$

Now that we have all the information we need we can do this in R.



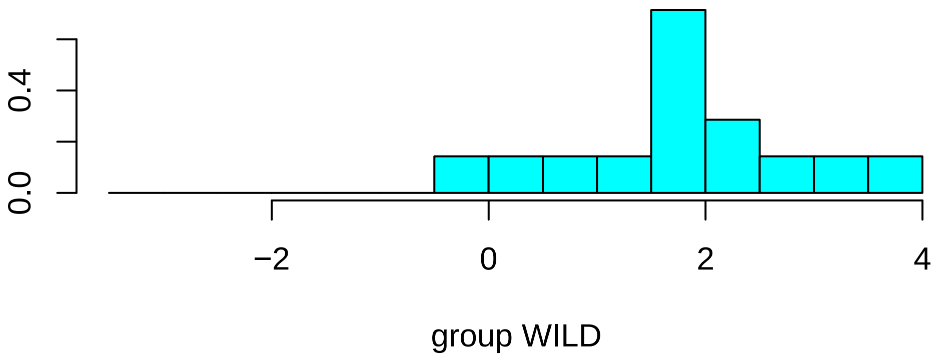


Figure 7: Histograms for the predicted value according to the first (and only in this case) linear discriminant

```
> tur.nmd.W <- tur.nmd[tur.nmd$TYPE == "WILD",]
> tur.nmd.D <- tur.nmd[tur.nmd$TYPE == "DOMESTIC",]
> tur.nmd.W.m <- apply(tur.nmd.W[,-c(1:3)],2,mean)
> tur.nmd.D.m <- apply(tur.nmd.D[,-c(1:3)],2,mean)
> S.1 <- var(tur.nmd.W[,-c(1:3)])
> S.2 <- var(tur.nmd.D[,-c(1:3)])
> n.1 <- dim(tur.nmd.W)[1]
> n.2 <- dim(tur.nmd.D)[1]
> S.pooled <- (n.1 - 1)/(n.1 + n.2 -2) * S.1 +
+ (n.2 - 1)/(n.1 + n.2 -2) * S.2

> round(S.pooled)
      HUM RAD ULN FEMUR TIN CAR D3P COR SCA
HUM    17   9  14    15  14  48  18  15  12
RAD     9  13  11    10  12  49  15   9   7
ULN    14  11  18    15  15  40  15  15  12
FEMUR  15  10  15    22  16  26  16  14   9
TIN    14  12  15    16  24  32  17  11   8
CAR    48  49  40    26  32 514 174  50  43
D3P    18  15  15    16  17 174 159  28  16
COR    15   9  15    14  11  50  28  22  14
SCA    12   7  12     9   8  43  16  14  18

> X <- as.matrix(tur.nmd[,-c(1:3)])
> a.hat <- (tur.nmd.W.m - tur.nmd.D.m)%*%solve(S.pooled)
> pred.y <- as.matrix(tur.nmd[,-c(1:3)])%*%t(a.hat)
> m.hat <- 0.5 * (tur.nmd.W.m - tur.nmd.D.m) %*% solve(S.pooled)
+ %*% (tur.nmd.W.m + tur.nmd.D.m)
> m.hat
         [,1]
[1,] 14.57087

# The following package contains useful functions for plotting
> library(lattice)
> densityplot(pred.y,groups=tur.nmd$TYPE,cex=1.5,
+ panel=function(x,y,...){
+ panel.densityplot(x,...)
+ panel.abline(v=14.57087,...)
+ }
+ )
```

Figure 8 shows the density and the vertical line which determines the allocating rule. This value was calculated as 14.57. Notice that one of each turkeys falls in the incorrect region. So we made two mistakes in total.

```
> tur.lda <- lda(tur.nmd[,4:12],tur.nmd[,3])
```

```
## Just in case you didn't do it before
> tur.ld <- predict(tur.lda)
> table(tur.nmd$TYPE,tur.ld$class)

           DOMESTIC WILD
  DOMESTIC       18    1
  WILD            1   13
```

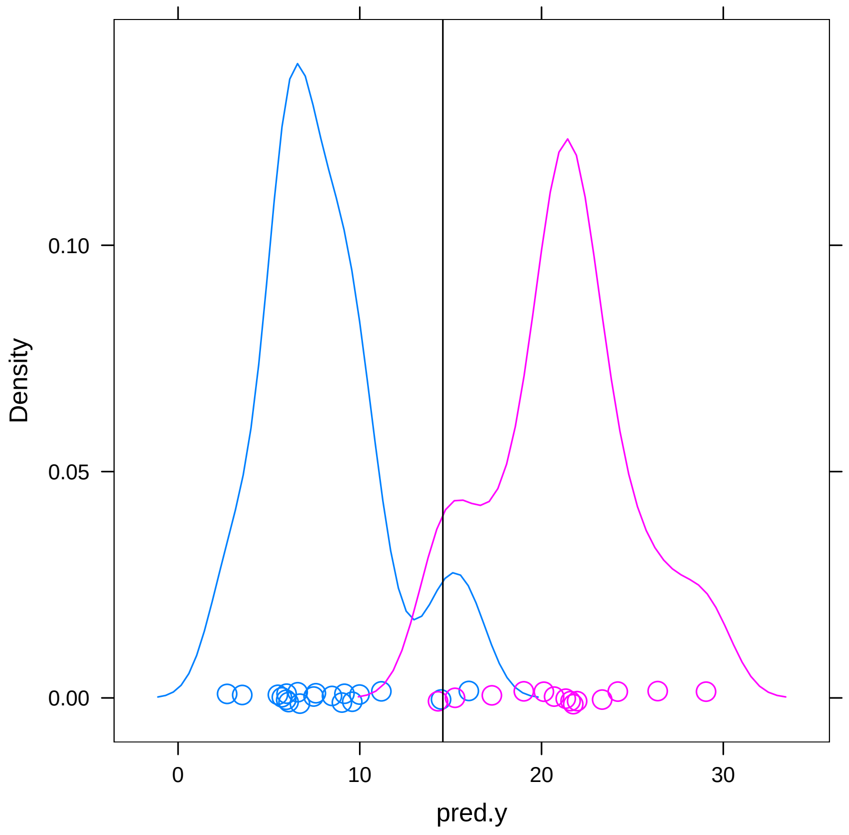


Figure 8: Density plot of the distribution of domestic and wild turkeys. The vertical line represents the boundary for deciding whether we allocate an individual to 'domestic' or 'wild'. Wild turkeys tend to have values larger than 14.57 and domestic smaller values than 14.57 (i.e. scores)

In GGobi you can explore the separation among the two groups with the 2-D tour. Make sure you *paint* the two types of turkeys with different colors and then you can explore projections which separate the groups using the projection pursuit and selecting LDA. For "interesting" projections select the optimize option.

6.1 *Polistes* wasps example–more than two groups

A researcher was interested in discriminating among four behavioral groups of primitively eusocial wasps from the genus *Polistes* based on gene expression data measured in the brain (based on qrt-PCR). The data was *log* transformed before the analysis.

Figure 9: Example of *Polistes* nest queens and workers.

The four groups are: foundress (F), worker (W), queens (Q), and gynes (G). The gene expression data was based on 32 genes. First we need to remove the missing data. Then we load the MASS package and use the lda function.

```
> wasp2 <- read.table("wasp.txt",header=T,na.strings=".")
> wasp.nmd <- na.omit(wasp2)
> library(MASS)
> wasp.lda <- lda(wasp.nmd[,-1],wasp.nmd[,1])
> wasp.lda
Call:
lda(wasp.nmd[, -1], wasp.nmd[, 1])

Prior probabilities of groups:
   F    G    Q    W
0.26 0.22 0.27 0.26
... [most output deleted]
Proportion of trace:
   LD1    LD2    LD3
0.5714 0.3234 0.1051
> table(true=wasp.nmd$grp,pred=predict(wasp.lda)$class)
    pred
```

```
true  F  G  Q  W
   F 18  0  0  1
   G  0 16  0  0
   Q  0  0 20  0
   W  2  0  0 17
```

The previous table shows the very optimistic estimate of our error rate. We have misclassified 3 wasps out of 74 (0.041). At this point this is evidence that the discrimination task is promising, but we should still try to estimate the true error rate through cross-validation. The following method is equivalent to holding out one individual and then predicting its membership from the classification rule based on the information of the remaining individuals.

```
> wasp.lda.cv <- lda(wasp.nmd[,-1],wasp.nmd[,1],CV=T)
> table(true=wasp.nmd$grp,pred=wasp.lda.cv$class)
    pred
true  F  G  Q  W
   F 12  0  2  5
   G  1 11  3  1
   Q  1  0 16  3
   W  6  1  3  9
```

In this case the error rate has increased to 0.35 in total but notice that we have been able to discriminate some groups better than others. Now let us produce a plot of the LD scores for LD1 and LD2 discriminants.

```
> wasp.ld <- predict(wasp.lda,dimen=2)$x
> eqscplot(wasp.ld,type="n",xlab="first linear discriminant",
+  ylab="second linear discriminant")
> text(wasp.ld, labels=as.character(wasp.nmd$grp),
+ col=1+unclass(wasp.nmd$grp),cex=0.8)
```

We can also identify which individuals where the ones that were misclassified by looking at the posterior probabilities. But first let us look at the `predict` function.

```
> wasp.ld <- predict(wasp.lda)
> names(wasp.ld)
[1] "class"     "posterior" "x"
```

This provides the scores (`x`), the predicted class and the posterior probabilities.

```
> wasp.pred <- data.frame(true=wasp.nmd$grp,pred=wasp.ld$class,
+ round(wasp.ld$post,2))
> wasp.pred[wasp.pred$true != wasp.pred$ pred,]
   true pred    F G    Q    W
8     F    W 0.32 0 0.06 0.62
35    W    F 0.52 0 0.00 0.48
38    W    F 0.65 0 0.00 0.35
```

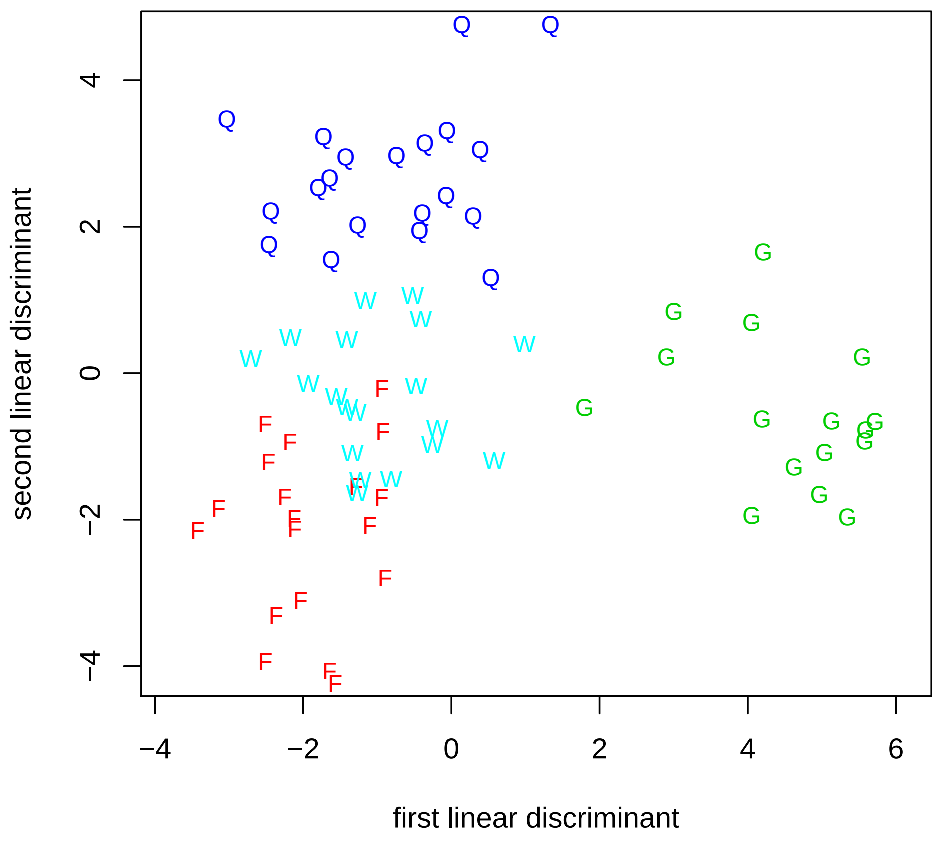


Figure 10: Linear discriminant scores for the `wasp2` dataset for LD1 and LD2. Queens (Q), Workers (W), Foundresses (F), and Gynes (G).

We have identified the three wasps which were incorrectly classified in the "plug-in" or "resubstitution" method.

Performing discriminant analysis without assuming that the variance-covariance matrices of each group are similar enough to be pooled requires the use of *quadratic* discriminant analysis. This is done with the `qda` function, also in the `MASS` package.

```
> wasp.qda <- qda(wasp.nmd[,-1],wasp.nmd[,1])
Error in qda.default(x, grouping, ...) : some group is too small for qda
```

In this case the number of observations per group is small. As a result, the respective covariance matrices are singular and it is not reasonable to conduct a quadratic discriminant analysis. We will stick with the LD which gave satisfactory results.

7 Logistic Regression

Among the methods proposed to solve the "supervised classification" problem extensions for traditional linear models are possible. The class textbook proposes *logistic regression* as an alternative to linear discriminant. Logistic regression is normally used when the variable of interest falls in one of two categories. For example, infected or not infected, dead or alive, etc. Logistic regression is one example of a *generalized linear model.* Notice the difference between *generalized* and *general.* In SAS linear models are fitted using the GLM procedure and these models were called *general linear models* because they generalize the approach in ANOVA and linear regression. In R these latter models are just linear models and can be fitted using the `lm` function. However, models which do not necessarily assume a normally distributed response variable are called *generalized* linear models. These latter models are fitted by GENMOD in SAS and by the `glm` function in R. The textbook shows a more specialized procedure called LOGISTIC.

In R I suggest trying `?glm` and investigate.

7.1 Crabs example

The example here is taken from Agresti (2002). In this example female crabs have "satellites" (or males) and we would like to know if we can predict the presence or absence of satellites from a set of variables measured on the females. The data set has a variable `satell` which indicates the number of crabs. But for this example we are only interested in the presence or absence.

```
> crabs <- read.table("crabs2.txt",header=T)
> names(crabs)
[1] "color"  "spine"  "width"  "satell" "weight" "dark"
```

Let us create a new variable which indicates satellite presence. I will use the function `attach` which makes the variables readily available but I discourage the use of this function except for this kind of short simple manipulation.

```
> attach(crabs)
> crabs$satpre <- ifelse(satell > 0 , 1, 0)
> summary(crabs$satpre)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  1.0000  0.6416  1.0000  1.0000
> class(crabs$satpre)
[1] "numeric"
> detach(crabs)
```

I used the `ifelse` function. The first argument is the condition, the second is the value if the condition is true and the third argument if it is not true. There is additional useful information and finally I detach `crabs`.

For a binomial response variable we can use the `glm` function with the `binomial` option for the `family` argument. The default link is the *logit*.

```
> fit.glm <- glm(satpre ~ factor(color) + factor(spine) + width + weight ,
+ data = crabs,family=binomial())
```

The `ifelse` function comes in handy again for handling the output.

```
> preds <- ifelse(fit.glm$fitted.values < 0.5,0,1)
> table(true=crabs$satpre,pred=preds)
    pred
true  0  1
   0 32 30
   1 15 96
```

We can also test for significance.

```
> summary(fit.glm)
Call:
glm(formula = satpre ~ factor(color) + factor(spine) + width +
    weight, family = binomial(), data = crabs)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1977  -0.9424   0.4849   0.8491   2.1198
Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -8.06501    3.92855  -2.053   0.0401 *
factor(color)2 -0.10290    0.78259  -0.131   0.8954
factor(color)3 -0.48886    0.85312  -0.573   0.5666
factor(color)4 -1.60867    0.93553  -1.720   0.0855 .
factor(spine)2 -0.09598    0.70337  -0.136   0.8915
factor(spine)3  0.40029    0.50270   0.796   0.4259
width           0.26313    0.19530   1.347   0.1779
weight          0.82578    0.70383   1.173   0.2407
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 225.76  on 172  degrees of freedom
Residual deviance: 185.20  on 165  degrees of freedom
AIC: 201.2
Number of Fisher Scoring iterations: 4
```

Now I decided to simplify the crabs example and use only two variables `weight` and `spine`.

```
> fit.glm.p <- predict(fit.glm,type="response")
> crabs$preds <- fit.glm.p
> attach(crabs)
> crabs.o <- crabs[order(weight),]
> xyplot(preds ~ weight , groups=spine , data=crabs.o,type="l",
+ lwd=3,auto.key=T)
```

Also in the `lattice` package we can find the endlessly useful `xyplot` function.

7.2 Turkey example revisited

The `turkey` example had only two groups (i.e. `WILD` and `DOMESTIC`) so this is a problem that could potentially be tackled by logistic regression. As with any regression problem which can potentially include many variables it is wise to first look at the correlation among the variables because of the well known collinearity problem in multiple regression. For this I'm introducing a bit more advanced graphics in R using again the `lattice` package. I'm not going to lie to you here, R graphics take a long time to master and sometimes to figure out something relatively simple you need to spend patient hours reading the documentation. The advantage is that as far as I know virtually anything is possible (notice I didn't say easy). If you intend to make graphics for publications many formats are available. Notice that I'm reusing the `tur.nmd` object we created in a previous session.

```
> library(lattice)
> splom(tur.nmd[,4:12],groups=tur.nmd$TYPE,pch=16,
+ col=c("blue","orange") , cex=0.8, varname.cex=0.7,pscales=0)
```

I chose `blue` and `orange` but I realize these might not be your favorite colors. Change them as you please.

Figure 12 shows that several variables are good candidates to produce a separation between `WILD` and `DOMESTIC` turkeys, the problem is that these variables are highly correlated. The problem of correlated variables in multiple regression is called 'collinearity' (or 'multicolinearity') and if you want to know more about this you can find the details in any regression textbook.

To keep the example easy we will include only the variable `TIN`.

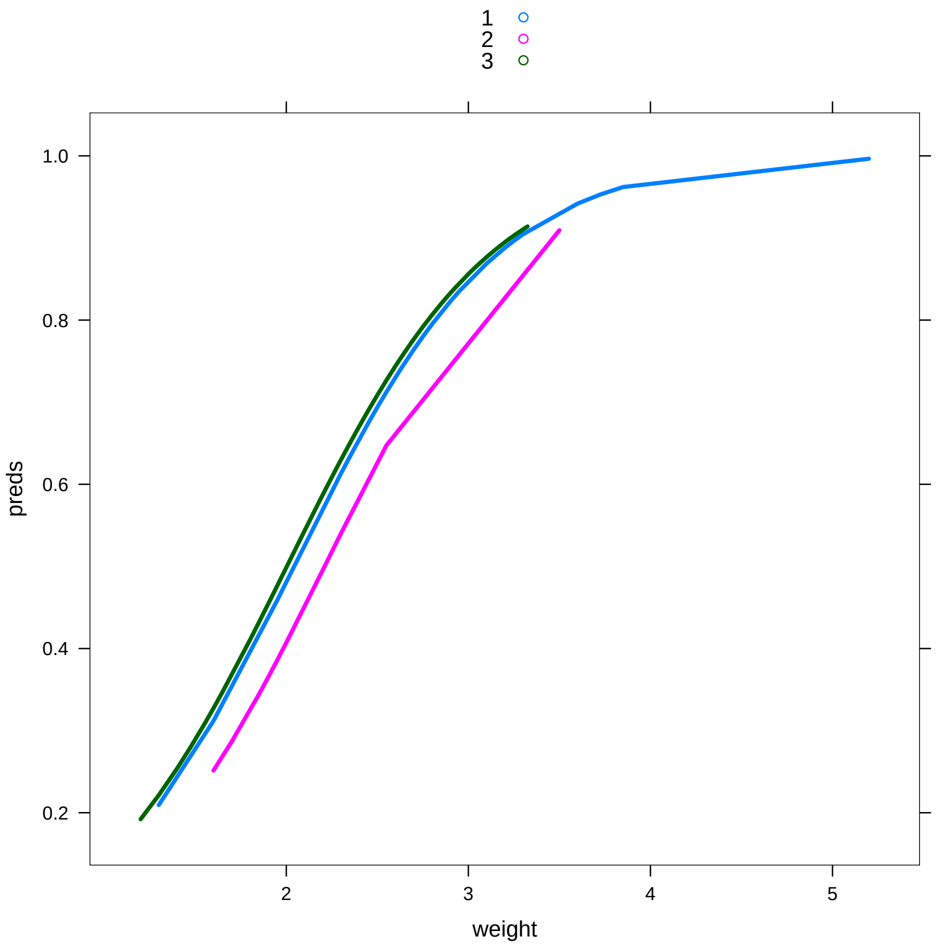


Figure 11: Predicted probability for presence of satellites for the `crabs` dataset using variables `spine` and `weight`

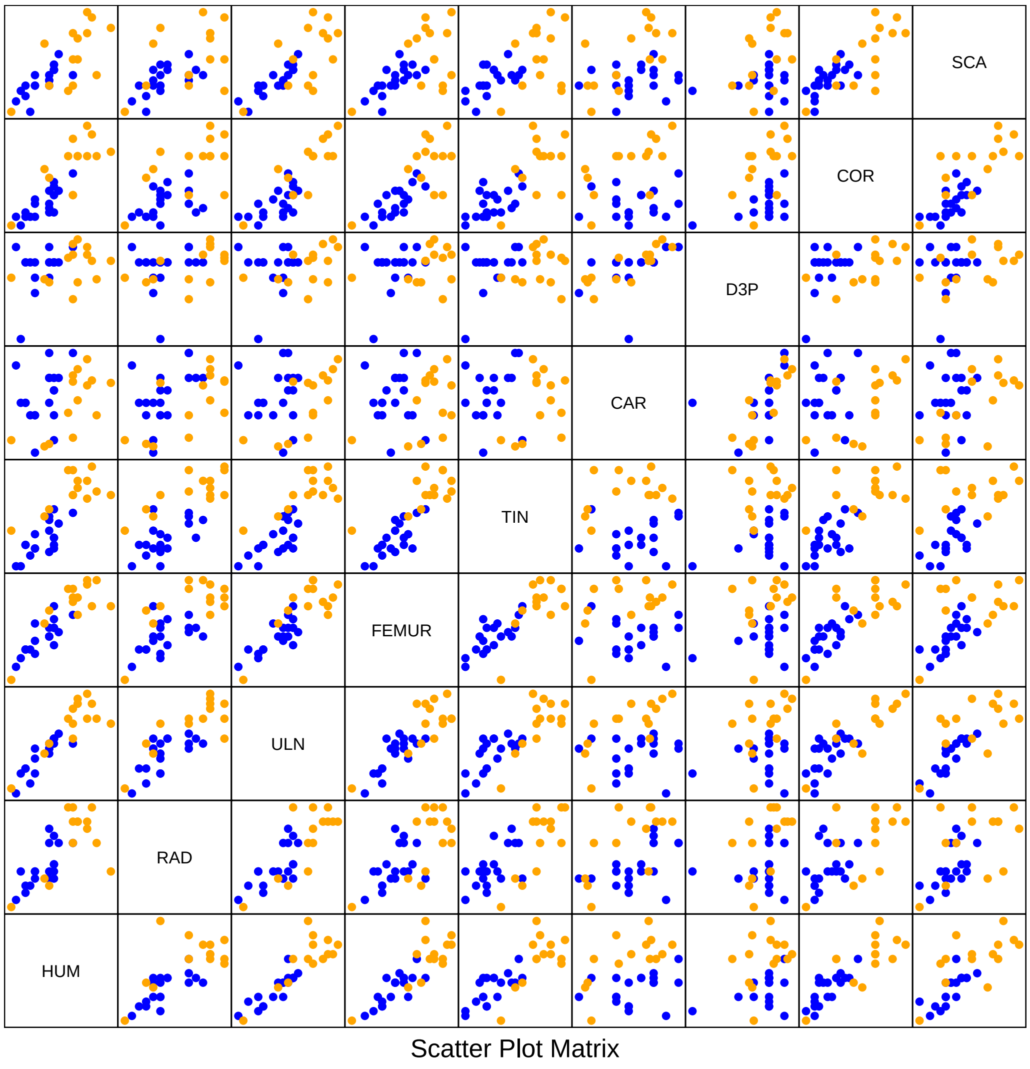


Figure 12: Scatterplot matrix of the variables in the `turkey` example. The two colors represent the two types of turkeys (WILD and DOMESTIC).

```
> fit.tur.glm <- glm(TYPE ~ TIN , data = tur.nmd ,
+ family=binomial("logit"))
> anova(fit.tur.glm , test="Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: TYPE
Terms added sequentially (first to last)
     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    32     44.987
TIN   1   30.199        31     14.788 3.899e-08
```

Without getting into too much detail we can see that the variable `TIN` is significant.

8 Cluster analysis

Discovering ‘clusters’ or groups of individuals which are similar within a group but different from other groups is mainly an exploratory exercise. GGobi is an excellent tool for this purpose. Let me quote Venables and Ripley,

> Do not assume that ‘clustering’ methods are the best way to discover interesting groupings in the data; in our experience the visualization methods are often far more effective. There are many different clustering methods, often giving different answers, and so the danger of over-interpretation is high.

8.1 Dataset `pluton` in the `cluster` package

This is a dataset taken from the `cluster` package.

8.1.1 Description

The `pluton` data frame has 45 rows and 4 columns, containing percentages of isotopic composition of 45 Plutonium batches.

8.1.2 Format

This data frame contains the following columns:

- Pu238: the percentages of (238)Pu, always less than 2 percent.
- Pu239: the percentages of (239)Pu, typically between 60 and 80 percent (from neutron capture of Uranium, (238)U).
- Pu240: percentage of the plutonium 240 isotope.
- Pu241: percentage of the plutonium 241 isotope.

8.1.3 References

Rousseeuw, P.J. and Kaufman, L and Trauwaert, E. (1996) Fuzzy clustering using scatter matrices, Computational Statistics and Data Analysis 23(1), 135151.

8.2 using R and GGobi

8.2.1 Distance

```
> library(cluster)
> data(pluton)
> pairs(pluton)
```

Clustering methods usually rely on distances among the *units* or *observations*. In R we can calculate distances using the `dist` function. The available methods are:

Method

```
The distance measure to be used. This must be one of "euclidean",
 "maximum", "manhattan", "canberra", "binary" or "minkowski".
Any unambiguous substring can be given.
```

Notice that many functions in R accept an *unambiguous substring.*

We will use the default euclidean method.

```
> d.pluton <- dist(pluton, method="eu")
> round(d.pluton,2)
> d.pluton2 <- dist(pluton, method="eu", diag=T, upper=T)
> round(as.matrix(d.pluton2),1)[1:10,1:10]
      1    2    3    4    5    6    7    8    9   10
1   0.0  0.4  0.8  4.1  3.3  4.3  0.1  0.6 13.8 17.9
2   0.4  0.0  0.4  4.5  3.2  4.6  0.3  0.2 13.5 17.5
3   0.8  0.4  0.0  4.9  3.0  5.1  0.8  0.2 13.1 17.1
4   4.1  4.5  4.9  0.0  6.3  0.3  4.2  4.7 17.5 21.5
5   3.3  3.2  3.0  6.3  0.0  6.5  3.2  3.1 11.3 15.3
6   4.3  4.6  5.1  0.3  6.5  0.0  4.3  4.9 17.7 21.8
7   0.1  0.3  0.8  4.2  3.2  4.3  0.0  0.6 13.8 17.8
8   0.6  0.2  0.2  4.7  3.1  4.9  0.6  0.0 13.3 17.3
9  13.8 13.5 13.1 17.5 11.3 17.7 13.8 13.3  0.0  4.0
10 17.9 17.5 17.1 21.5 15.3 21.8 17.8 17.3  4.0  0.0
```

8.3 *K*-means

This is an example of K-means which attempts to separate the observations into clusters. I will choose 3 clusters because they are pretty clear after we have done the work, but before it is not always this obvious.

```
> plut.km <- kmeans(pluton , 3)
> plut.km
K-means clustering with 3 clusters of sizes 18, 12, 15
Cluster means:
      Pu238    Pu239    Pu240    Pu241
1 0.1561111 76.59300 20.19483 2.312222
2 0.7430833 71.57317 19.97392 5.685417
3 1.3830667 60.63393 24.38753 8.666467
[...]
Within cluster sum of squares by cluster:
[1]  75.87455 122.78870  54.80182
Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

I suppressed some of the output. The part I suppressed is in fact important and I will use it below to evaluate the clusters. We can combine the results from `kmeans` with the original dataset and explore it in GGobi.

```
> plut2 <- data.frame(pluton,clust=plut.km$cluster)
> gplut2 <- ggobi(plut2)
```

Notice that in this case the clusters are probably not the optimum solution since some observations were assigned to a cluster which has a large gap between them.

It might be better at least to try 10 different starts,

```
>  plut.km2 <- kmeans(pluton , 3, nstart=10)
>  plut.km2
K-means clustering with 3 clusters of sizes 15, 24, 6
Cluster means:
      Pu238    Pu239    Pu240    Pu241
1 1.3830667 60.63393 24.38753 8.666467
2 0.2119583 75.68517 20.50300 2.659250
3 1.1066667 70.18467 18.52033 7.670500
[...]
Within cluster sum of squares by cluster:
[1]  54.80182 153.91551  22.45733
Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

Let us look at the data again in GGobi and see the clusters more clearly.

We can also produce a nice scatterplot matrix using the `lattice` package.

```
> library(lattice)
> splom(pluton , groups = plut.km2$cluster,cex=1.2,pch=c(16,21,22))
```

It is fairly easy to identify the clusters using GGobi.

```
gpluton <- ggobi(pluton)
```

Use the 2-D tour to paint the clusters and identify them.

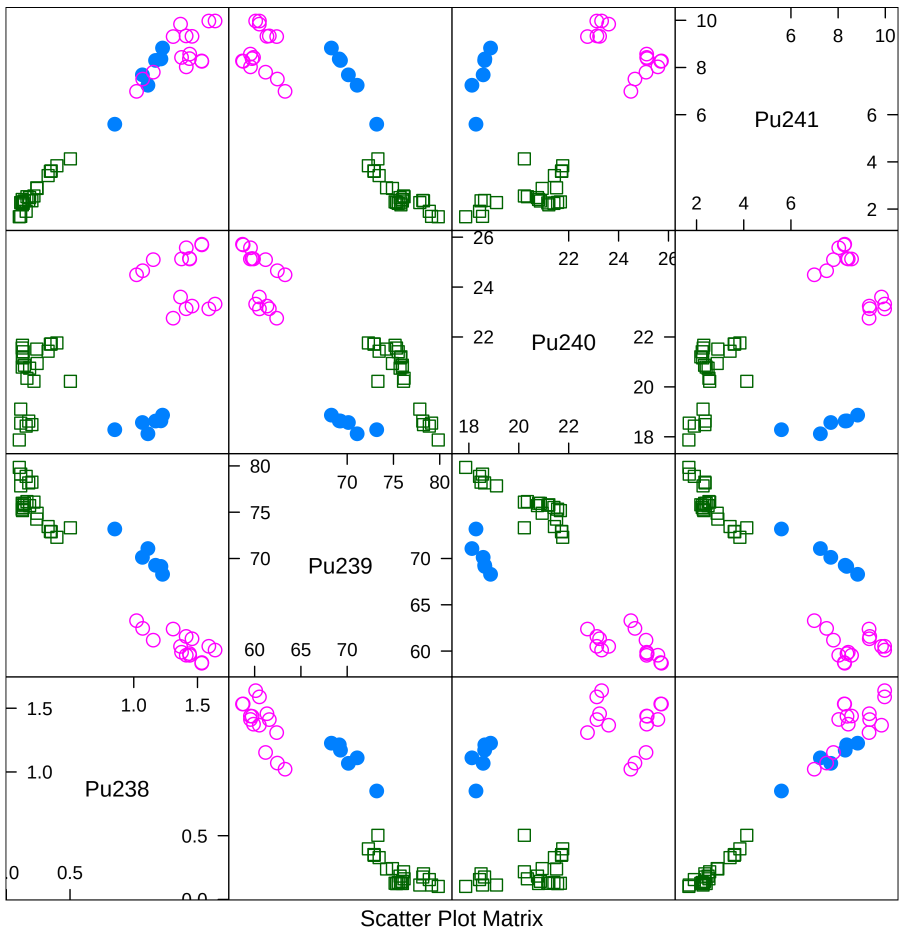


Figure 13: Scatterplot matrix of `pluton` dataset with the clusters determined by `kmeans` using 10 initial starting points.

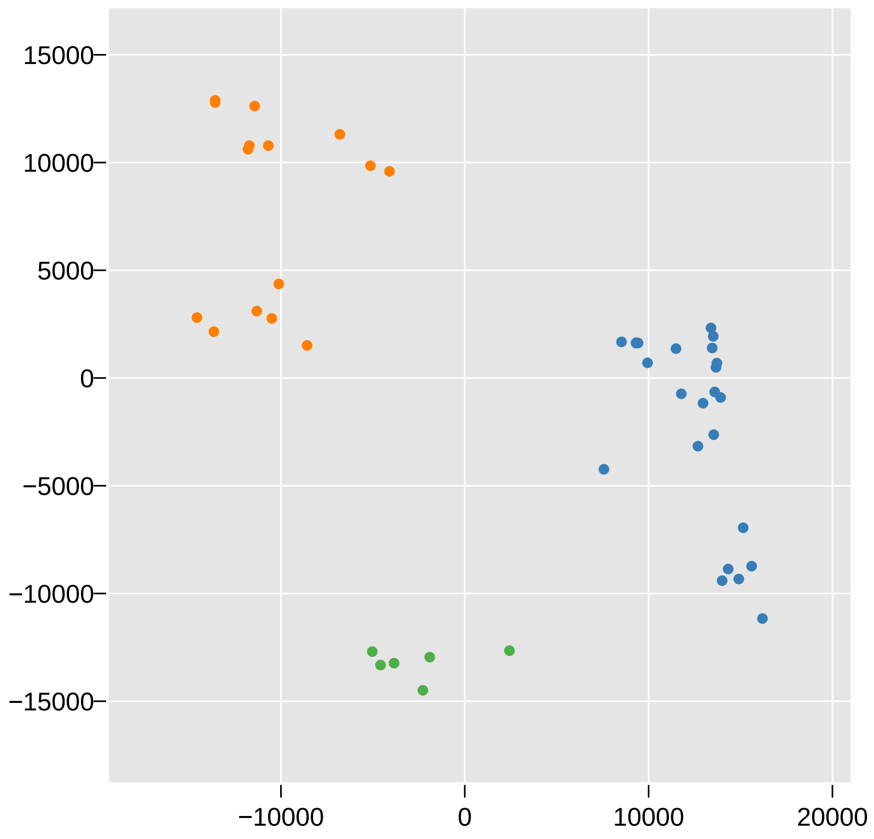


Figure 14: This graph was produced using the `DescribeDisplay` package for GGobi. This is a projection from the 2-D tour.

8.4 Pizza example

This example is taken from the textbook. First we can perform a K-means analysis but this time we will do a little programming to explore different number of clusters

```
pizza <- read.table("pizza.txt",header=T,na.strings=".")
pizza.nmd <- na.omit(scale(pizza[,2:8]))
n <- dim(pizza.nmd)[1]
wss1 <- (n-1)*sum(apply(pizza.nmd,2,var))
wss <- numeric(9)
for(i in 2:10){
   W <- sum(kmeans(pizza.nmd,i)$withinss)
   wss[i-1] <- W
}
wss <- c(wss1,wss)
plot(1:10,wss,type="l",xlab="Number of groups",
ylab="Within groups sum of squares",lwd=2)
```

The previous code requires some explanation. First I omit the missing values. Then I calculate the number of rows and I stored them in an object called `n`. The object `wss1` contains the total within sums of squares. The object `wss` is an empty vector which will be used to store the results of the `kmeans`. Then, I'm using a new element in R: the `for` function which initializes a loop. The value of `i` will cycle from 2 to 10. This value is latter used in the `kmeans` function as an argument indicating the number of clusters. The object `W` stores the sum of the within sums of squares of the clusters. When we exit the loop we have an object `wss` which has collected the within sums of squares for choices of 2 through 10 clusters. Finally, we plot.

It is interesting to perform an analysis in GGobi for the pizza dataset. We can also examine the results from the `kmeans` by doing

```
> pizza.km4 <- kmeans(pizza.nmd, 4, nstart=10)
> pizza2 <- data.frame(pizza.nmd,clust=pizza.km4$clust)
> gpizza <- ggobi(pizza2)
```

One possible separation of the clusters produced Figure 16.

Another example of using programming tools in R is shown using hierarchical clustering of the `pizza` dataset.

```
pizza.nmd <- na.omit(scale(pizza[,2:8]))
pizza.d <- dist(pizza.nmd)
## We will choose only these four methods
methods <- c("ward","single","complete","average")
par(mfrow=c(2,2))
for(i in 1:4){
    meth <- methods[i]
    h <- hclust(pizza.d , method=meth)
```

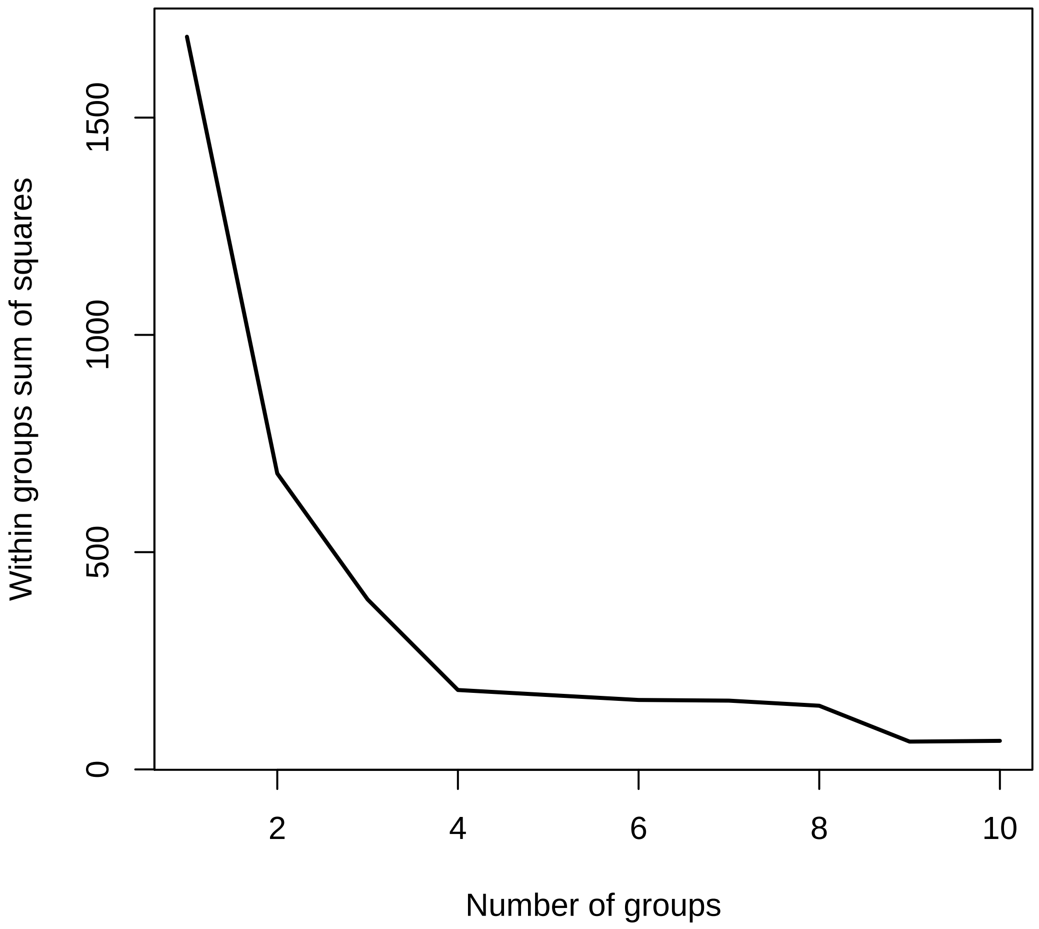


Figure 15: Within sums of squares for different choices of number of clusters using the `kmeans` method for the `pizza` dataset.

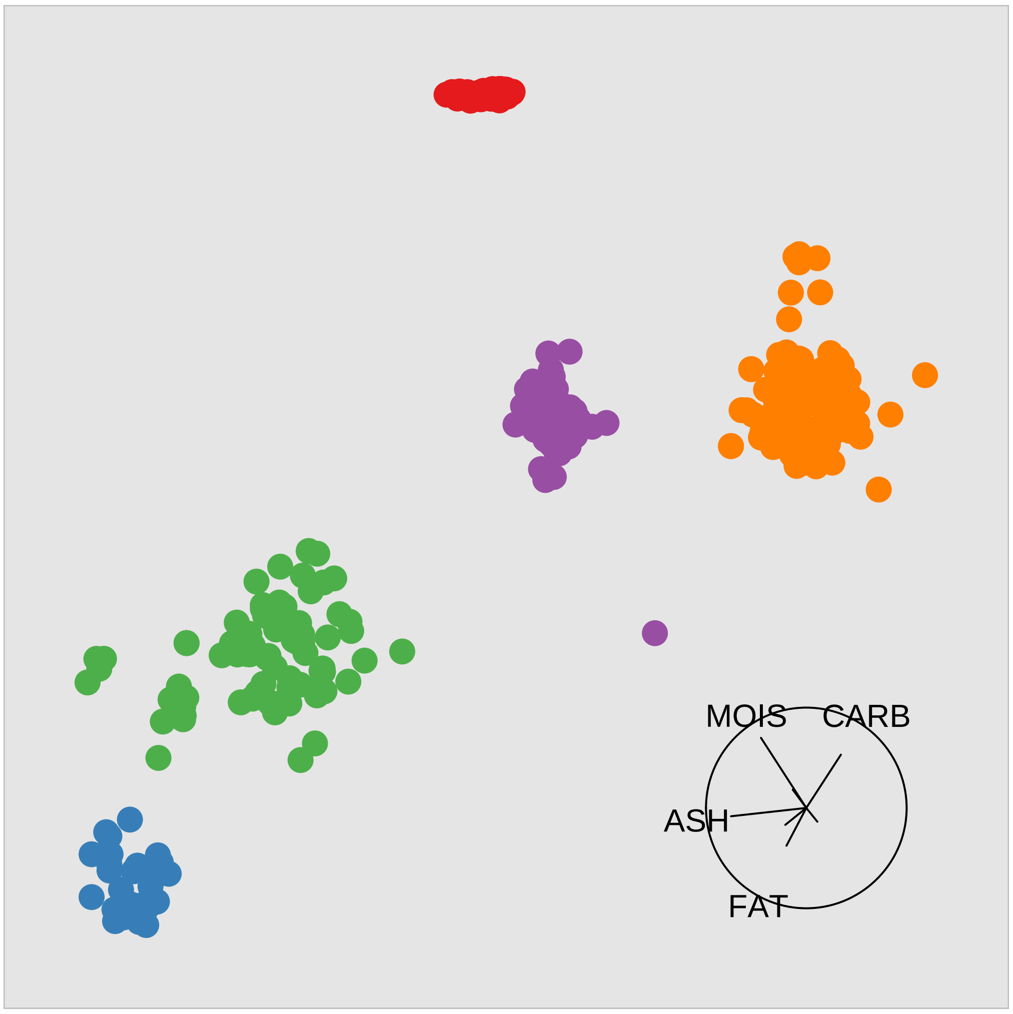


Figure 16: Two dimensional projection of the `pizza` dataset using GGobi. Five main clusters were identified in this case.

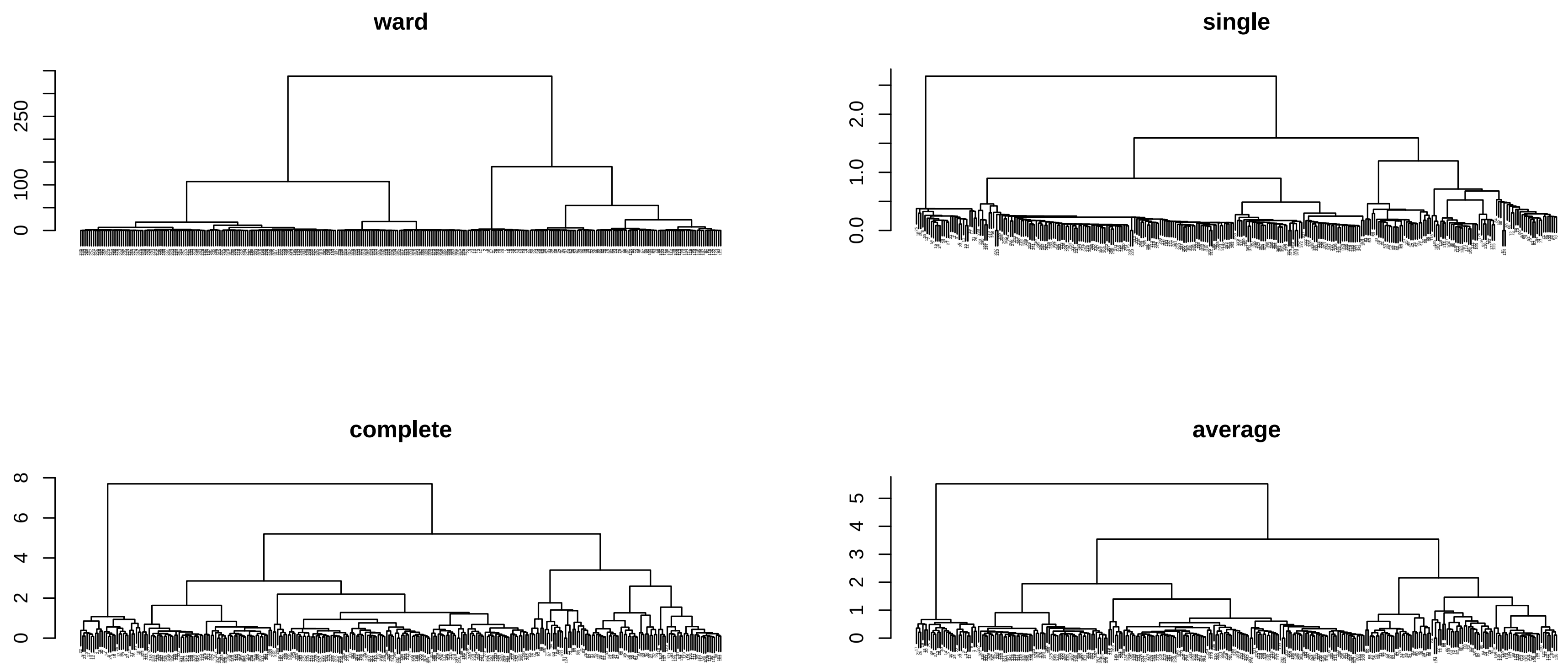


Figure 17: Four dendrograms for the different clustering methods on the `pizza` dataset

```
    plot(h, ann=F,cex=0.2)
    title(meth)
}
```

First I created a vector with the four selected methods, then I set up the plotting device with four panels. Then I do a loop over the methods by picking the method by its index in the vector.

Let us examine the results from the "single" method but we will choose four clusters.

```
h.s <- hclust(pizza.d, method="single")
h.s4 <- cutree(h.s,k=4)
```

The cutree function comes in handy to cut the tree at four clusters. We could have also specified the height where we want to 'cut' the tree.

It is quite informative to examine the results in GGobi.

```
pizza2 <- data.frame(pizza.nmd,clust=h.s4)
library(rggobi)
gpizza <- ggobi(pizza2)
```

In this example I think that GGobi is a much more powerful tool to discover clusters. The 2-D projection shown in Figure 16 shows that 5 clusters might be more appropriate. In any case the choice of the number clusters would depend on the specific problem, but in GGobi we can examine in detail the outliers and also identify observations, which is extremely useful in real life.

8.5 More pizza

Let us go back to the pizza dataset and start from ground zero. In general when we explore multivariate datasets is useful to start with summary statistics, then one dimensional plots and then build to more complex plots.

```
> round(cor(na.omit(pizza[,2:8])),2)
        MOIS  PROT   FAT   ASH SODIUM  CARB   CAL
MOIS    1.00  0.36 -0.17  0.27  -0.10 -0.59 -0.76
PROT    0.36  1.00  0.50  0.82   0.43 -0.85  0.07
FAT    -0.17  0.50  1.00  0.79   0.93 -0.64  0.76
ASH     0.27  0.82  0.79  1.00   0.81 -0.90  0.33
SODIUM -0.10  0.43  0.93  0.81   1.00 -0.62  0.67
CARB   -0.59 -0.85 -0.64 -0.90  -0.62  1.00 -0.02
CAL    -0.76  0.07  0.76  0.33   0.67 -0.02  1.00
```

One student pointed out that calories are calculated as a function of PROT, FAT and CARB. The formula implies that we could back calculate FAT as $FAT = (CAL \times 100 - PROT \times 4 - CARB \times 4)/9$. So the missing data is hard to justify. It is possible

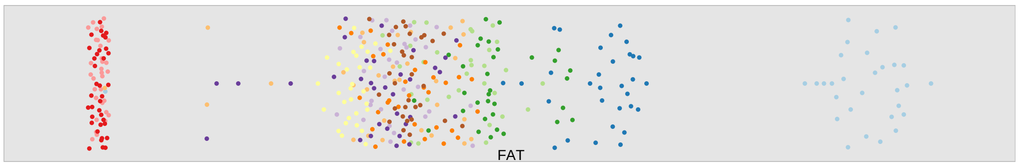


Figure 18: One dimensional texturized dotplot of `FAT` variable. Notice that this variable alone separates three groups and the groups correspond approximately to distinct `BRANDS`.

that the pizza brands which have missing data for `FAT` claim to be “fat-free”, but this should mean that the correct value would be zero and not missing.

One dimensional plots of calories, fat and other variables reveal a lot of information (Fig 18). For example, the variable fat alone separates three groups of pizza.

A clustering using `kmeans` of the `pizza` dataset could group data into BRANDS nicely

```
> table(brand=pizza.nmd$BRAND,clus=pizza.km5$cluster)
     clus
brand  1  2  3  4
    A  0  0 29  0
    B 31  0  0  0
    C 27  0  0  0
    D 32  0  0  0
    E  0  0  0  0
    F  0  0  0  0
    G  0 28  0  0
    H  0 30  0  0
    I  0 29  0  0
    J  0 33  0  0
    K  0  0  0 29
    L  0  0  0 32
```

9 Mean vectors and Covariance matrices

Formal methods for multivariate analysis are, of course, available. Although they are not used as much as the previous methods they are nevertheless useful and it is important to understand the mechanics in order to better understand the previous methods. We will start this section by constructing confidence regions for a sample with two variables. Johnson and Wichern (2002) says that

A 100(1-α)% *confidence region* for the mean of a p-dimensional normal distribution is the ellipsoid determined by all $\boldsymbol{\mu}$ such that,

$$n(\bar{\boldsymbol{x}} - \boldsymbol{\mu})' \boldsymbol{S}^{-1} (\bar{\boldsymbol{x}} - \boldsymbol{\mu}) \leq \frac{p(n-1)}{(n-p)} F_{p,n-p}(\alpha)$$

where

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{x_j}, \quad \boldsymbol{S} = \frac{1}{(n-1)}\sum_{j=1}^{n}(\boldsymbol{x}_j - \bar{\boldsymbol{x}})(\boldsymbol{x}_j - \bar{\boldsymbol{x}})'$$

and

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$$

are the sample observations.

This implies that beginning at the center $\bar{\mathbf{x}}$, the axes of the confidence ellipsoid are

$$\pm\sqrt{\lambda_i}\sqrt{\frac{p(n-1)}{n(n-p)}F_{p,n-p}(\alpha)}\mathbf{e}_i$$

where

$$\boldsymbol{S}\boldsymbol{e}_i = \lambda_i \boldsymbol{e}_i, \quad i = 1, 2, \ldots, p$$

9.1 Soils example

The study aims to identify sensible properties at different depths within the soil profile that are able to reflect the improved effect of no-till corn-soybean rotations involving cover crops on soil properties compared with no-till corn-soybean rotations with winter fallowing. Seven soil properties were determined under 4 crop sequences and at 3 different depths of the soil. Therefore the present data set contains 2 classification variables, sequence and depth, with 4 and 3 levels respectively. The sequence ('seq') is corn-soybean rotations with either winter fallow ('fallow') or a winter cover crop of rye ('rye'), hairy vetch ('vetch'), or their mixture ('mix'). The depth is given in ranges, from 0 to 5 cm ('0-5cm'), 5 to 10 cm ('5-10cm'), and 10 to 15 cm ('10-15cm'). There are 7 dependent variables or soil properties recorded; soil reaction (pH), available phosphorus (P), soil organic matter (SOM), soil bulk density (DB), water aggregate stability (WAS), total nitrogen (TN), and penetration resistance (RP). There is a total of 288 observations and the treatments are balanced with 24 observations for each sequence at each recorded depth.

```
##No need to run this code. It is shown here to illustrate manipulations.
soil <- read.csv("alldataforFer.csv",header=T)
soil$logTN <- log(soil$TN)
soil$logNO3 <- log(soil$NO3)
soil2 <- soil[,-c(15,16)]
SoilV <- scale(soil2[,10:17])
soilD <- data.frame(SoilV,cover=soil2$covercrop)
## Run this code
soil <- read.csv("soils.csv",header=T)
```

Here we have taken the *log* of a couple of variables, we have eliminated the untransformed variables and we have scaled (or standardized) the variables to put them on equal footing. From now on we will only use the `soil` object. Let us plot the first 20 observations but selecting variables soil organic matter `SOM` and bulk density `DB`.

```
source("C:/LabWorkspace/ManovaT.R")
## we need some extra functions which are stored in the previous file
soilD.s <- soil[1:20,3:4]
 plot(soilD.s)
 points(ellipse(soilD.s),col="yellow")
```

The `ellipse` function seems magical but we could do some work to show the principles.

```
n <- nrow(soilD.s)
p <- ncol(soilD.s)
pF.crit <- (n-1)*p * qf(0.95,p,n-p)/(n*(n-p))
X.var <- var(soilD.s)
X.m <- apply(soilD.s,2,mean)
eval <- eigen(X.var)$val
evec <- eigen(X.var)$vec
y0 <- sqrt(pF.crit * eval[1]) * evec[,1] + X.m
y1 <- sqrt(pF.crit * eval[2]) * evec[,2] + X.m


##Code for plotting the confidence ellipse
 plot(soilD.s,xlim=c(-2,2.5),ylim=c(-2,2))
 points(ellipse(soilD.s),col="yellow",cex=0.8,pch=".")
 points(ellipse(soilD.s,cl=0.99),col="blue",cex=0.8,pch=".")
arrows(X.m[1],X.m[2],y0[1],y0[2],col="orange",lwd=2,code=0)
arrows(X.m[1],X.m[2],y1[1],y1[2],col="orange",lwd=2,code=0)
```

We can also use the `ggobiManova` function to investigate the data.

```
gsoil <- ggobiManova(soil[,1:8],soil[,9])
```

9.2 MANOVA in R

In R it is possible to perform multivariate analysis of variance (MANOVA) using the `manova` function. Most of the time MANOVA is applied on data coming from a desgned experiment. The `soil` data set is a good example for this method.

```
soil <- read.csv("alldataforFer.csv",header=T)
soil.mnv <- manova(as.matrix(soil[,10:17]) ~ city + factor(block)
+ factor(year) + depth*covercrop, data=soil)
names(soil.mnv)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "contrasts"     "xlevels"       "call"          "terms"
[13] "model"
```

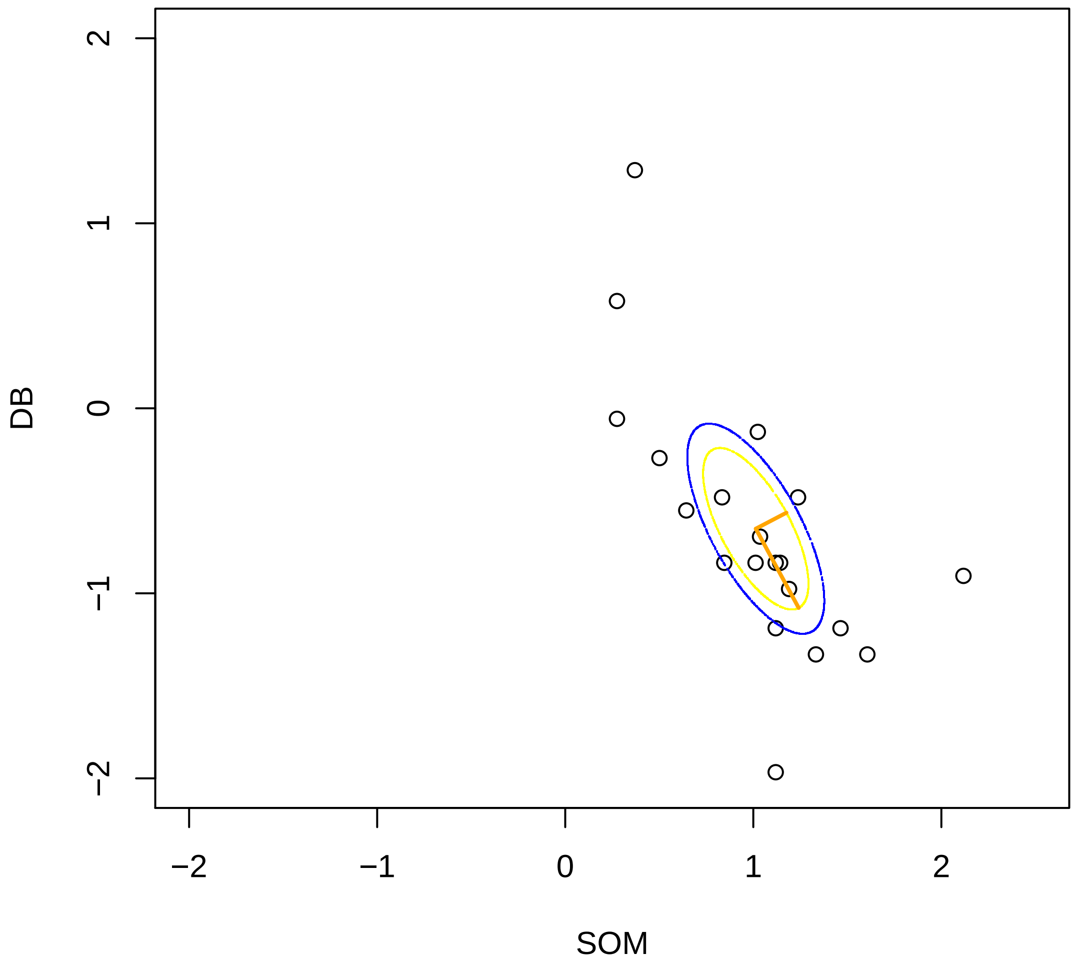


Figure 19: Bulk density (DB) and soil organic matter (SOM) selection from the `soilD` dataset which illustrates the determination of confidence regions. The yellow ellipse is the 95 % confidence ellipse and the blue is the 99% confidence ellipse. The orange lines are the half-axes of the ellipse corresponding to the 95 % ellipse. Why did I choose these limits for the x and y axis?

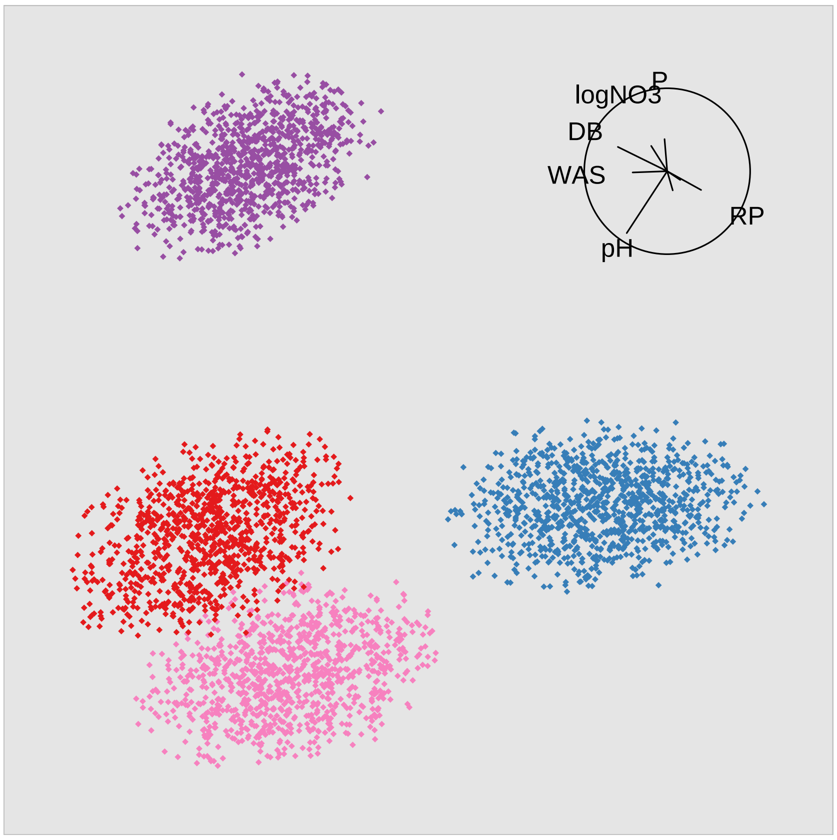


Figure 20: 95 % Confidence ellipses for the four cover crops. Purple is ‘fallow’, blue is ‘vetch’, pink is ‘mix’, and red is ‘rye’.

Quite a few objects are stored within the `soil.mnv`, which is the result of performing MANOVA on the `soil` data. Formal tests can be obtained with the `summary` function.

```
> summary(soil.mnv)
                  Df  Pillai approx F num Df den Df    Pr(>F)
city               1   0.755  101.723      8    264 < 2.2e-16 ***
factor(block)      3   0.374    4.731     24    798 1.481e-12 ***
factor(year)       1   0.687   72.597      8    264 < 2.2e-16 ***
depth              2   1.030   35.172     16    530 < 2.2e-16 ***
covercrop          3   0.297    3.659     24    798 1.177e-08 ***
depth:covercrop    6   0.109    0.622     48   1614    0.9805
Residuals        271
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The default test is Pillai but this can be changed in the `test` argument. In this case the fact that depth by covercrop is not significant seems to be a good result because this means that cover crops are not behaving differently at different depths. However, there is an effect of `depth` and `covercrop`.

By extracting the residuals, we can easily obtain the sums of squares cross product matrix for the errors (SSCP) and partial correlations.

```
## Sums of squares cross products matrix for the errors.
> SSCP.E <- crossprod(soil.mnv$residuals)
> round(SSCP.E)
       pH     P   SOM   DB   WAS  TN    NO3       RP
pH     82  -341  -152    4   418   1   -955    -1954
P    -341 26415  4048  -72 -2531  44    241    38584
SOM  -152  4048  5122  -40  -734  17   3292    -9854
DB      4   -72   -40    3    32   0   -101     1942
WAS   418 -2531  -734   32 20567 -36  -9592    94823
TN      1    44    17    0   -36  27    -70      144
NO3  -955   241  3292 -101 -9592 -70 107519   -11628
RP  -1954 38584 -9854 1942 94823 144 -11628 24795288
## Partial Correlations
pCC.E <- cor(soil.mnv$residuals)
round(pCC.E,2)
       pH     P   SOM    DB   WAS    TN   NO3    RP
pH   1.00 -0.23 -0.23  0.26  0.32  0.01 -0.32 -0.04
P   -0.23  1.00  0.35 -0.27 -0.11  0.05  0.00  0.05
SOM -0.23  0.35  1.00 -0.34 -0.07  0.05  0.14 -0.03
DB   0.26 -0.27 -0.34  1.00  0.13 -0.03 -0.19  0.24
WAS  0.32 -0.11 -0.07  0.13  1.00 -0.05 -0.20  0.13
TN   0.01  0.05  0.05 -0.03 -0.05  1.00 -0.04  0.01
NO3 -0.32  0.00  0.14 -0.19 -0.20 -0.04  1.00 -0.01
```

```
RP  -0.04  0.05 -0.03  0.24  0.13  0.01 -0.01  1.00
```

Next I will show a plot which scales all variables from 0 to 1 and then it joins with a line the same row or 'observation'. It is called a parallel coordinates plot. We need some additional code.

```
soil$depth <- factor(soil$depth,levels=c("0-5cm","5-10cm","10-15cm"))
```

The previous code is useful when we have variables which are not correctly ordered by `R` and we want to redefine the order of the levels. This is useful to then plot in GGobi.

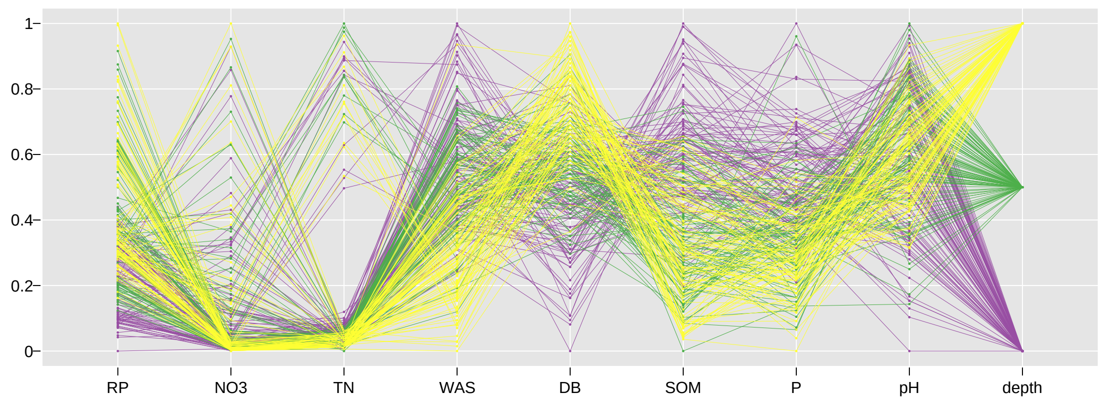


Figure 21: Parallel coordinates plot for the soil data colored by depth.

9.3 Effluent data

We are interested in comparing measurements made by two labs, one commercial and the other one a state laboratory. In the effluent samples two variables were measured biochemical oxygen demand (`BOD`) and suspended solids (`SS`). These data were taken from Johnson and Wichern (2002).

```
labs <- read.table("JW6-1.txt",header=T)
## Make sure the ManovaT.R file is sourced
 labs
    Lab BOD  SS
1   Comm   6  27
2   Comm   6  23
3   Comm  18  64
4   Comm   8  44
5   Comm  11  30
6   Comm  34  75
7   Comm  28  26
```

```
8   Comm  71 124
9   Comm  43  54
10  Comm  33  30
11  Comm  20  14
12 State  25  15
13 State  28  13
14 State  36  22
15 State  35  29
16 State  15  31
17 State  44  64
18 State  42  30
19 State  54  64
20 State  34  56
21 State  29  20
22 State  39  21
m.diff <- labs[1:11,-1] - labs[12:22,-1]
```

A formal test can be conducted for this example.

```
> d.bar <- colMeans(m.diff)
> S.d <- var(m.diff)
> S.d
> T.sq <- 11 * t(d.bar)%*%solve(S.d)%*%d.bar
> T.sq
         [,1]
[1,] 13.63931
> qf(0.95,2,9)*(20/9)
[1] 9.458877
```

Now we need to compare the T^2 to the critical point. Taking $\alpha = 0.05$ we find that $[p(n-1)/(n-p)]F_{p,n-p}(.05) = [2(10)/9]F_{2,9}(.05) = 9.47$. Since $T^2 = 13.6 > 9.47$ we reject H_o.

The 95 % simultaneous confidence intervals for the mean differences δ_1 and δ_2 can be computed,

$$\delta_1 : \bar{d}_1 \pm \sqrt{\frac{(n-1)p}{(n-p)}F_{p,n-p}(\alpha)}\sqrt{\frac{s^2_{d_1}}{n}}$$

$$-9.36 \pm \sqrt{9.47}\sqrt{\frac{199.26}{11}}$$

$$(-22.46, 3.74)$$

and

$$13.27 \pm \sqrt{9.47}\sqrt{\frac{418.61}{11}}$$

$$(-5.71, 32.25)$$

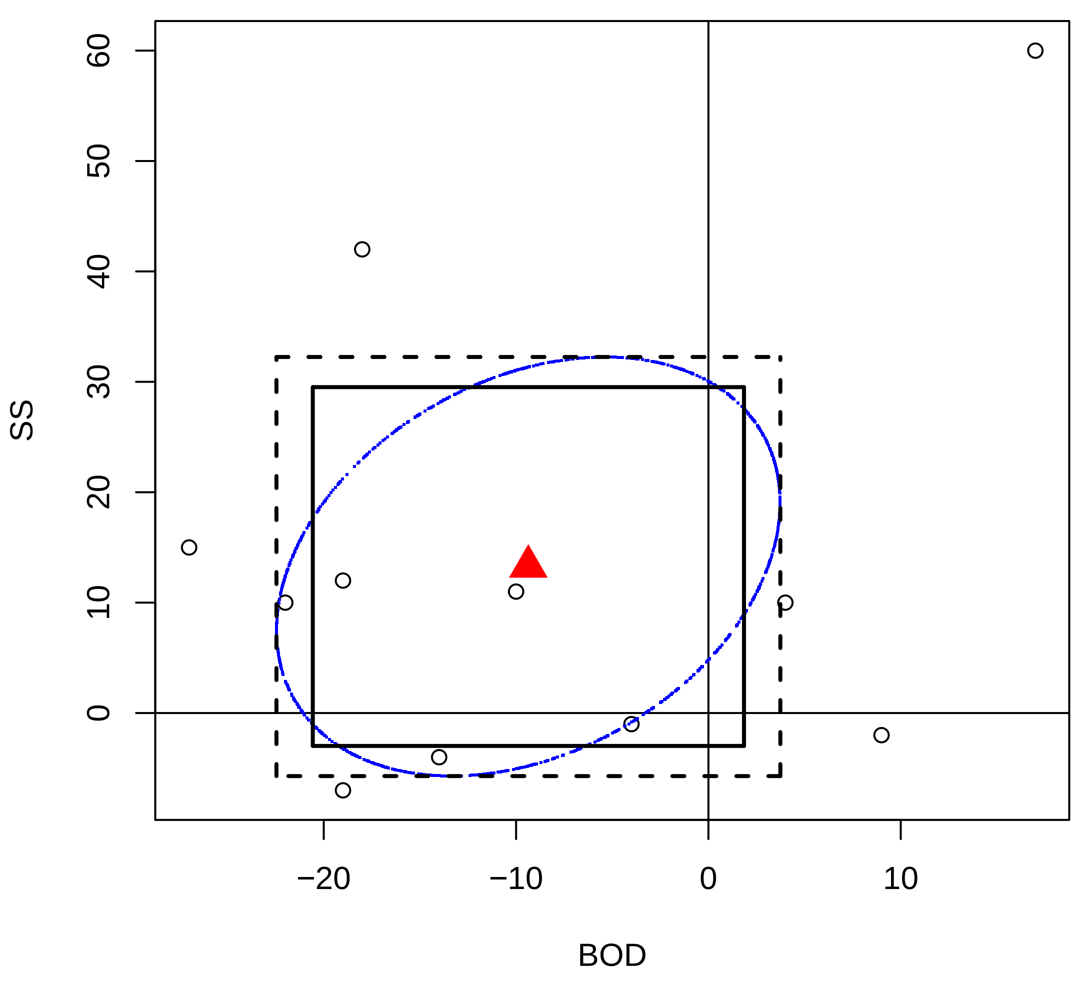


Figure 22: Differences between the two variables for the Effluent data. The ellipse is the 95 % confidence region for the paired differences of means. The solid lines are the Bonferroni confidence intervals and the dashed line are the simultaneous confidence intervals.

For the bonferroni intervals we simply use $t_{n-1}\left(\frac{\alpha}{2p}\right)$. So,

$$\delta_1 : \bar{d}_1 \pm t_{n-1}\left(\frac{\alpha}{2p}\right)\sqrt{\frac{s^2_{d_1}}{n}}$$

$$-9.36 \pm 2.63\sqrt{\frac{199.26}{11}}$$

$$(-20.57, 1.85)$$

and

$$13.27 \pm 2.63\sqrt{\frac{418.61}{11}}$$

$$(-2.98, 29.52)$$

```
> arm <- qt(1-(0.05/(2*2)),10) * sqrt(199.26/11)
> c(-9.36 - arm, -9.36 + arm)
[1] -20.569624   1.849624

> arm2 <- qt(1-(0.05/(2*2)),10) * sqrt(418.61/11)
> c(13.27 - arm2, 13.27 + arm2)
[1] -2.977472 29.517472
```

The code for the plot

```
plot(m.diff)
abline(h=0,v=0)
points(ellipse(m.diff),pch=".",col="blue")
points(d.bar[1],d.bar[2],pch=2,col="red",cex=3)
## Simultaneous intervals
arrows(3.74,-5.71,3.74,32.25,lty=2,code=0)
arrows(3.74,-5.71,-22.46,-5.71,lty=2,code=0)
arrows(-22.46,-5.71,-22.46,32.25,lty=2,code=0)
arrows(-22.46,32.25,3.74,32.25,lty=2,code=0)
## Bonferroni
arrows(1.85,-2.98,1.85,29.52,lty=1,code=0)
arrows(1.85,-2.98,-20.57,-2.98,lty=1,code=0)
arrows(-20.57,-2.98,-20.57,29.52,lty=1,code=0)
arrows(-20.57,29.52,1.85,29.52,lty=1,code=0)
```

This example is particularly interesting because although the individual shadows include zero the 95 % confidence region does not include the $(0, 0)$ coordinate. It appears that the commercial lab tends to produce lower BOD and higher SS measurements than the state lab.

10 Canonical Correlations

Canonical Correlation Analysis seeks to find relationships between two sets of variables. In particular we try to find linear combinations of the first set of variables which have maximum correlation with the second set of variables. The idea is first to determine the pair of linear combinations having the largest correlation. Next, we determine the pair of linear combinations having the largest correlation among all pairs uncorrelated with the initially selected pair, and so on.

In R canonical correlation analysis can be performed with the `cancor` function, but note that it is simple enough to do a canonical correlation analysis by hand (i.e. using matrix functions). Probably because of its generally limited use in providing a useful interpretation of the data, canonical correlation analysis is not developed much in the base R. Contributed packages provide many additional features that might be worth investigating if this method is needed.

10.1 Theory

Let us consider the data matrix $\boldsymbol{X}_{\boldsymbol{n \times p}}$ we can obtain the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{n \times p}}$. However, the structure of the data might suggest two distinct groups. For example, sometimes we might be taking measurements that fall in the 'chemical' category and an other set which fall in the 'physical' category. In taxonomical studies measurements can be divided in 'morphological' and 'molecular'. It might be of interest to examine possible relationships between these groups of observations. We can accommodate the variables in the following way,

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma_{11}} & \boldsymbol{\Sigma_{12}} \\ \boldsymbol{\Sigma_{21}} & \boldsymbol{\Sigma_{22}} \end{bmatrix}$$

The variance-covariance matrices on the diagonal represent the relationship within the two groups. The variance-covariance matrices on the off-diagonal represent the correlations between the variables in the first group with the variables on the second group. The previous arrangement contains all the information we need to perform canonical correlations. The objective is to find linear combinations in the first group that maximize the correlation with a linear combination in the second group.

$$\rho_1 = \max_{\boldsymbol{a \neq 0, b \neq 0}} [corr(\boldsymbol{a'x_1}, \boldsymbol{b'x_2})]$$

It turns out that the eigenvalues of

$$\boldsymbol{\Sigma_{11}^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}}$$

are the squared canonical correlations. The eigenvectors provide the linear combinations to produce the first two canonical variables for the first group. Similarly, the eigenvalues and eigenvectors for

$$\boldsymbol{\Sigma_{22}^{-1}\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}}$$

provide the same eigenvalues and eigenvectors for the second group. Therefore these are easy to obtain and this will be shown in the next example.

10.2 Head example

The question of interest here is whether there is a relationship between the head measurements for pair of sons.

Canonical correlations can be implemented for the following example

```
headsize <- read.table("head.txt",header=T)
head1 <- headsize[,1:2]
head2 <- headsize[,3:4]
## Calculate correlation matrices
R11 <- cor(head1)
R22 <- cor(head2)
R12 <- c(cor(head1[,1],head2[,1]),cor(head1[,1],head2[,2]),
         cor(head1[,2],head2[,1]),cor(head1[,2],head2[,2]))
R12 <- matrix(R12,ncol=2,byrow=T)
R21 <- t(R12)
## Construct the E1 and E2 matrices
E1 <- solve(R11)%*%R12%*%solve(R22)%*%R21
E2 <- solve(R22)%*%R21%*%solve(R11)%*%R12
##
E1
E2
##
eigen(E1)
eigen(E2)
##
sqrt(eigen(E1)$val)  # These are the correlations
## If we need to produce a plot
 score1 <- as.matrix(head1)%*%eigen(E1)$vec[,1]
 score2 <- as.matrix(head2)%*%eigen(E2)$vec[,1]
 plot(-score2,score1,pch=16,xlab="Score 2",ylab="Score 1")
```

An alternative way of getting the same result in R is as follows.

```
cancor(head1,head2)
scoresSon1 <- as.matrix(head1)%*%cancor(head1,head2)$xcoef
scoresSon2 <- as.matrix(head2)%*%cancor(head1,head2)$ycoef
plot(scoresSon2[,1],scoresSon1[,1],pch=16)
```

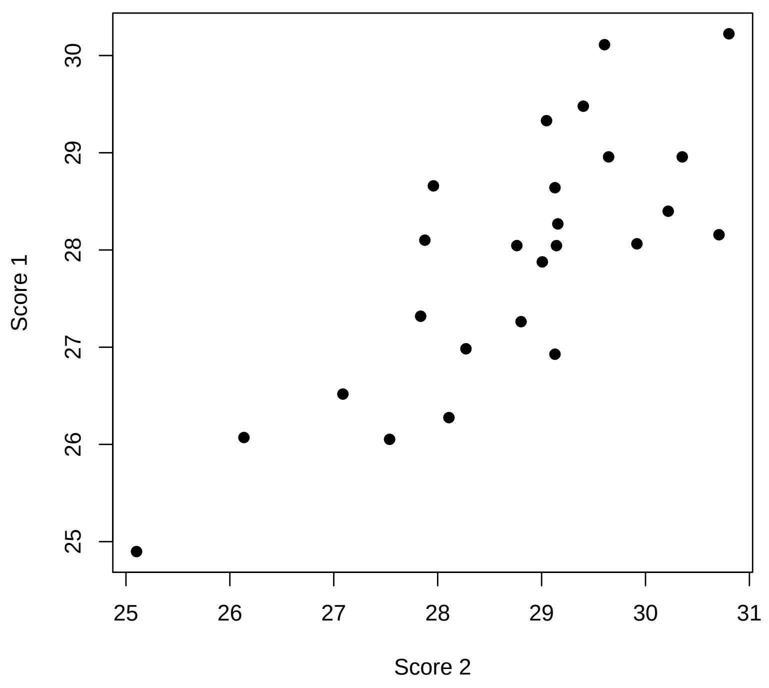


Figure 23: Canonical scores for the `head` example. The axis and the signs are arbitrary.

11 Missing data

Missing data is a problem often encountered by data analysts which examine multivariate data sets. Up to now we've been omitting missing data and while this is an acceptable approach when there is little missing data, it is not a reasonable approach when a large percentage of data are missing. This problem can get even worse when many variables are missing and we eliminate observations because only one of the variables measured is not available. The example used in the GGobi chapter about missing data is used here.

```
d.tao <- read.csv("tao.csv",header=T,row.names=1)
 gg <- ggobi(d.tao)
 g <- gg[1]
 glyph_type(g) <- 6
 glyph_size(g) <- 2
 gcolor = c(rep(1,368),rep(5,368))
 glyph_color(g) = gcolor
```

There are a few things to keep in mind when dealing with missing data.

- All missing data are not created equal.
- How much missing data?
- Are data missing at random or is there structure to the missing data?

Let us go back to the d.tao object. A simple summary indicates that there is missing data and it reports how many NA's per variable.

```
> summary(d.tao)
      Year         Latitude         Longitude       Sea.Surface.Temp
 Min.   :1993   Min.   :-5.000   Min.   :-110.0   Min.   :21.60
 1st Qu.:1993   1st Qu.:-2.000   1st Qu.:-110.0   1st Qu.:23.50
 Median :1995   Median :-1.000   Median :-102.5   Median :26.55
 Mean   :1995   Mean   :-1.375   Mean   :-102.5   Mean   :25.86
 3rd Qu.:1997   3rd Qu.: 0.000   3rd Qu.: -95.0   3rd Qu.:28.21
 Max.   :1997   Max.   : 0.000   Max.   : -95.0   Max.   :30.17
                                                  NA's   : 3.00
    Air.Temp        Humidity          UWind             VWind
 Min.   :21.42   Min.   :71.60   Min.   :-8.100   Min.   :-6.200
 1st Qu.:23.26   1st Qu.:81.30   1st Qu.:-5.100   1st Qu.: 1.500
 Median :24.52   Median :85.20   Median :-3.900   Median : 2.900
 Mean   :25.03   Mean   :84.43   Mean   :-3.716   Mean   : 2.636
 3rd Qu.:27.08   3rd Qu.:88.10   3rd Qu.:-2.600   3rd Qu.: 4.100
 Max.   :28.50   Max.   :94.80   Max.   : 4.300   Max.   : 7.300
 NA's   :81.00   NA's   :93.00
```

From the summary we can see that the two variables with the most missing values are `Air.Temp` (81) and `Humidity` (93). This is a lot of missing data and it is definitely worth investigating the nature of the missing values. Next we will follow the convention in GGobi and assign a 1 to a missing value and a 0 otherwise. In this way we can investigate if there is structure in the missing values. The function `is.na` is used to test if a value is missing or not.

```
> d.tao$Air.Miss <- ifelse(is.na(d.tao$Air.Temp),1,0)
> d.tao$Hum.Miss <- ifelse(is.na(d.tao$Humidity),1,0)
> table(d.tao$Year,d.tao$Air.Miss)
        0   1
  1993 364   4
  1997 291  77
> table(d.tao$Year,d.tao$Hum.Miss)
        0   1
  1993 275  93
  1997 368   0
```

We can see here that the missing data for `Air.Temp` occurred mostly in 1997 and that all of the missing values for `Humidity` occurred in 1993. This points to a definite structure in the nature of the missing data. We will continue the exploration of the missing data in GGobi.

Suppose we would like to replace all missing values in the `d.tao` object with zeros. Then this code should be used.

```
> d.tao[is.na(d.tao)] <- 0
> summary(d.tao)
      Year         Latitude          Longitude        Sea.Surface.Temp
 Min.   :1993   Min.   :-5.000   Min.   :-110.0   Min.   : 0.00
 1st Qu.:1993   1st Qu.:-2.000   1st Qu.:-110.0   1st Qu.:23.49
 Median :1995   Median :-1.000   Median :-102.5   Median :26.43
 Mean   :1995   Mean   :-1.375   Mean   :-102.5   Mean   :25.76
 3rd Qu.:1997   3rd Qu.: 0.000   3rd Qu.: -95.0   3rd Qu.:28.21
 Max.   :1997   Max.   : 0.000   Max.   : -95.0   Max.   :30.17
    Air.Temp        Humidity          UWind            VWind
 Min.   : 0.00   Min.   : 0.00   Min.   :-8.100   Min.   :-6.200
 1st Qu.:22.84   1st Qu.:79.00   1st Qu.:-5.100   1st Qu.: 1.500
 Median :24.07   Median :84.00   Median :-3.900   Median : 2.900
 Mean   :22.27   Mean   :73.77   Mean   :-3.716   Mean   : 2.636
 3rd Qu.:26.99   3rd Qu.:87.70   3rd Qu.:-2.600   3rd Qu.: 4.100
 Max.   :28.50   Max.   :94.80   Max.   : 4.300   Max.   : 7.300
```

Is this a good idea in this case? Almost surely, this is a bad idea in this case and performing multiple imputation conditional on the year is a much better idea. This is explored in more depth in the GGobi book chapter on missing data.

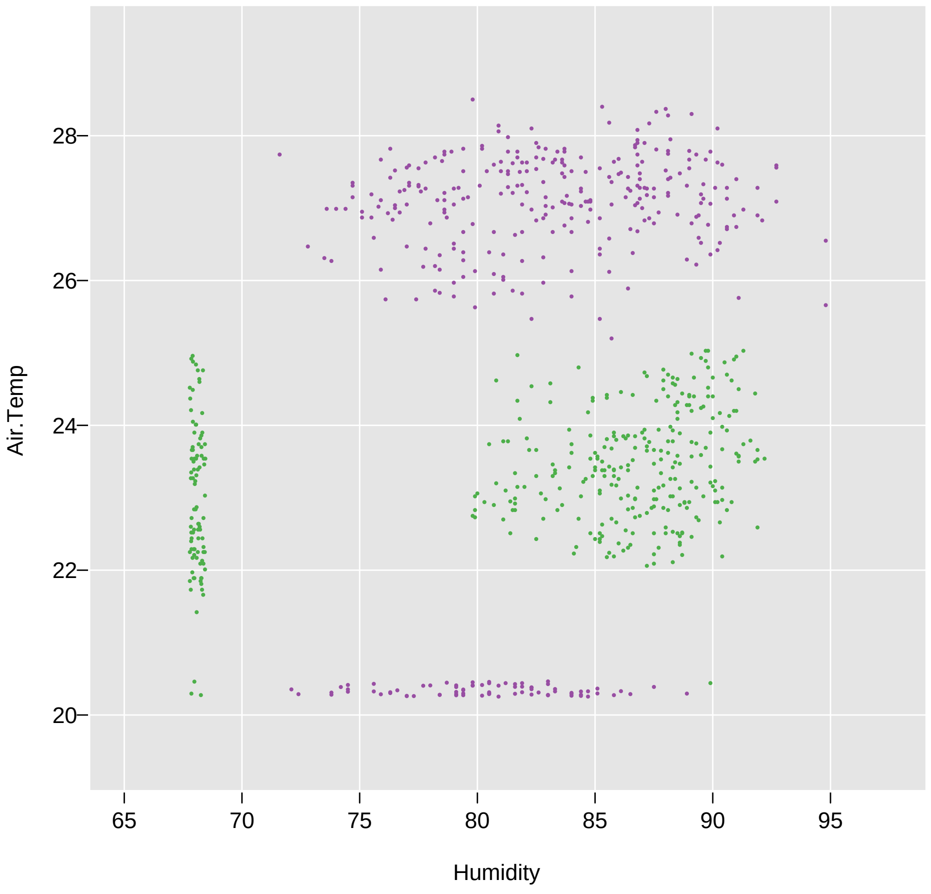


Figure 24: Scatter plot of `Air.Temp` and `Humidity` with missing data on the margins.

12 Concluding Remarks

This is the end of this manual for now. I hope it has provided an introduction to some of the methods for multivariate analysis as well as provide the basics for the use of R and GGobi. Multivariate analysis has a large exploratory component and most of the effort should be placed in the initial examination of the data. Simple descriptive statistics should be the first step. Usually, this first step identifies errors in the dataset and finding these errors early in the data analysis can prevent wasting time later at more advanced stages. Since data analysis should be largely an interactive process GGobi provides fantastic tools for this purpose. Now that this manual is over you must continue the study of multivariate statistical analysis on your own. There is so much to learn!

References

[1] A. Agresti. *Categorical Data Analysis.* Wiley, 2002.

[2] B. S. Everitt. *An R and S-plus companion for multivariate analysis.* Springer, 2005.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001.

[4] D.E. Johnson. *Applied multivariate methods for data analysts.* Duxbury Press, 1998.

[5] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis.* Prentice Hall, NJ, 5th edition, 2002.

[6] R. Khattree and D. N. Naik. *Applied multivariate statistics with SAS software.* SAS Institute, Cary, NC, Wiley, NY., 1999.

[7] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

Index