Statistics for Data Science (using R)

Javier Palarea-Albaladejo

University of Girona

Version 09/2023

Contents

Preface

These course notes provide an applied introduction to multivariate data analysis methods and statistical models using the R system for statistical computing. Currently, they are primarily aimed at students of the "Statistics for Data Science" course of the MSc in Data Science of the University of Girona and it serves as basis to more specialised courses taught later on. They build on previous materials developed by the author while delivering training courses for scientists at Biomathematics and Statistics Scotland (BioSS) and lecturing the Multivariate Data Analysis course at the University of Edinburgh. Basic statistical knowledge and some experience working and managing data in the R environment is assumed. The course avoids mathematical/statistical theory as much as possible and concentrates on the underlying concepts, emphasising how to put them in practice using R as computing tool.

They are divided into two blocks:

- Chapters 1-6: overview of some multivariate methods aimed at data dimension reduction, classification, identification of similarities, associations, and patters in data sets; with a focus on data exploration and graphical representation.
- Chapters 7-12: overview of some of the families of linear, non-linear, generalised linear and additive regression models commonly used in statistical modelling, including questions related to model validation, variable selection and dealing with high dimensions.

Some bibliographic references for more details include:

- Everitt, B. S. and Hothorn, T. (2011). *An Introduction to Applied Multivariate Analysis with R*. Springer.
- Faraway, J.J. (2005). *Linear Models with R*. Chapman & Hall/CRC, http://www.maths.bath.ac.uk/~jjf23/LMR.
- Faraway, J.J. (2006). *Extending the Linear Model with R*. Chapman & Hall/CRC, http://www.maths.bath.ac.uk/~jjf23/ELM.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R*. 2nd edition, Springer.
- Kuhn, M., and Johnson, K. (2013). *Applied predictive modeling*. Springer.
- Wood S.N. (2017) Generalized Additive Models: An Introduction with R (2nd edition). Chapman and Hall/CRC Press.
- Zelterman, D. (2015). *Applied Multivariate Statistics with R*. Springer.

The written material is accompanied by R code generating the outputs. We strongly encourage the students not to just copy and paste code chunks into their R system, but to type in and experiment with the code themselves, checking the details of the functions on the R help system, reviewing what arguments are available and how they interact with the output of the function. This will for sure provide a much more enriching learning experience.

An online version of these course notes can be found at https://imae.udg.edu/~jpalarea/StatsDS/.

1 Exploring the data

1.1 Introduction

As with any new data set it is important to organise multivariate data in a text file or spreadsheet before starting the formal analysis. The usual structure of the data file is a solid rectangle/matrix with column names and sample labels if available. Conventionally in the multivariate context we refer to having n samples/observations/individuals and p variables/features, commonly arranged by rows and columns respectively in a spreadsheet (called a `data.frame` in R). The data matrix denoted by $\mathbf{X}$ is formally

$$\mathbf{X} = \begin{pmatrix} x_{11} & . & . & x_{1j} & . & . & x_{1p} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ x_{i1} & . & . & x_{ij} & . & . & x_{ip} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ x_{n1} & . & . & x_{nj} & . & . & x_{np} \end{pmatrix}$$

Each row of $\mathbf{X}$ represents one sample. Each column of $\mathbf{X}$ represents one variable. The individual variables are here generically denoted $x_1, \ldots, x_p$. Of course, they may be other columns in the data set, for example factor or categorical variables (e.g. gender, region, type of tissue, …) and a second or third set of labels for the samples.

It is important to have a clear understanding of how you perceive the data. Which are the continuous variables, which the discrete and which the categorical? For example, the number of spots on a ladybird could be considered a discrete variable or a categorical variable. If considered categorical it might be thought of as ordered categorical, or *ordinal* variable, rather than just nominal (like e.g. gender or country of residence would be). Of course you might decide to change the way you think of a response as your analyses progress but it pays to start out with a thought about this.

Before employing any sophisticated statistical techniques it is also important to explore the data with appropriate graphical methods and to derive helpful summary statistics. The importance of data visualisation cannot be underestimated, particularly in a multivariate context.

1.2 The air pollution data set

In an attempt to link air pollution, as measured by sulphur dioxide, to weather and socio-geographical variables the following variables were recorded for 41 US cities.

- SO2 – sulphur dioxide (micrograms per cubic metre)
- Temp - Average annual temperature (Fahrenheit)
- Manuf - Number of Enterprises employing more the 20 people
- Pop – Population is thousands
- Wind – Average annual wind speed (mph)
- Precip - Average annual rainfall (inches)
- Days – Average number of days with rain per year

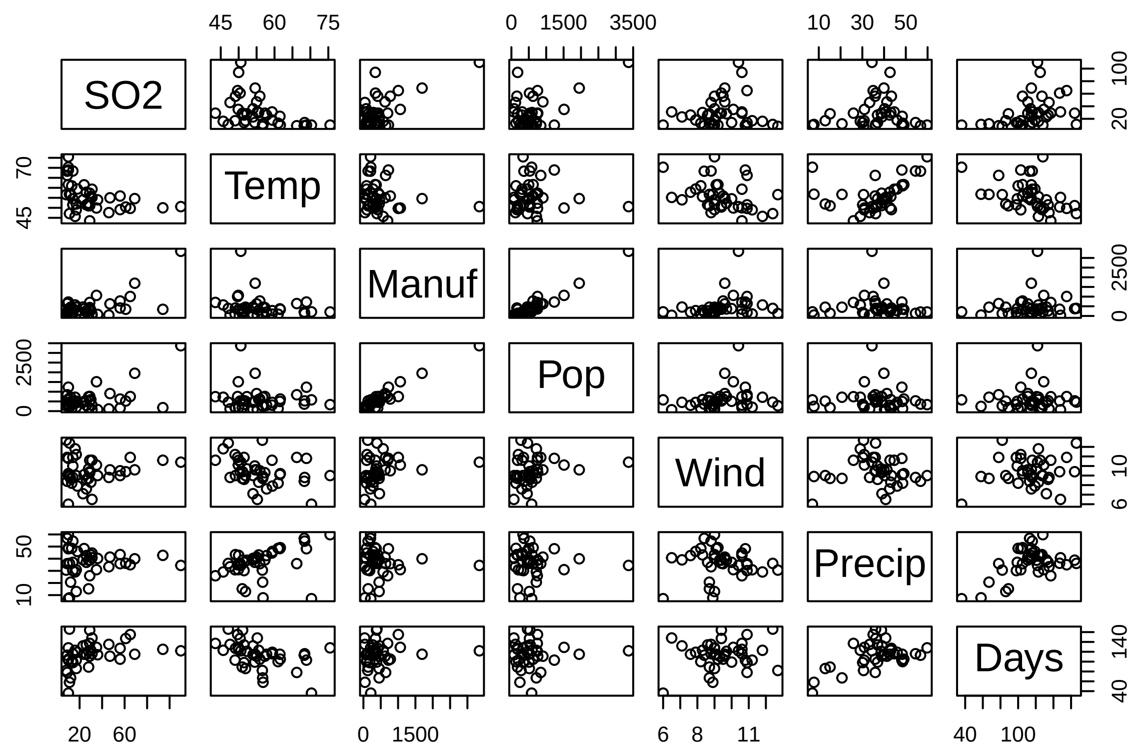
The following code imports the data set from the text file `air_pollution.txt` available in the `DataFiles` folder. Alternatively, the data set can be loaded directly from the copy in R format using `load("air_pollution.Rdata")`. Optionally the function `attach` can be used in R to be able to call the variables in the `data.frame` by name directly. This is fine as long as there are not several `data.frames` loaded on the R workspace including variables sharing the same name. Otherwise, variables in a `data.frame` are commonly accessed using the `$` operator (e.g. `airpol$Wind`) or by column number (e.g. `airpol[,7]`).

```r
# Import from the text file format
airpol <- read.table("air_pollution.txt", h = T, stringsAsFactors = TRUE)
# Useful to check the types of variables in a data.frame
str(airpol)
```

```
'data.frame':	41 obs. of  9 variables:
 $ Town  : Factor w/ 41 levels "Albany","Alburquerque",..: 31 20 36 12 15 41 39 18 23 3 ...
 $ Tn    : Factor w/ 41 levels "Al","Aq","At",..: 33 21 37 13 15 41 39 18 23 3 ...
 $ SO2   : int  10 13 12 17 56 36 29 14 10 24 ...
 $ Temp  : num  70.3 61 56.7 51.9 49.1 54 57.3 68.4 75.5 61.5 ...
 $ Manuf : int  213 91 453 454 412 80 434 136 207 368 ...
 $ Pop   : int  582 132 716 515 158 80 757 529 335 497 ...
 $ Wind  : num  6 8.2 8.7 9 9 9 9.3 8.8 9 9.1 ...
 $ Precip: num  7.05 48.52 20.66 12.95 43.37 ...
 $ Days  : int  36 100 67 86 127 114 111 116 128 115 ...
```

Before applying any formal statistical analysis is always convenient to explore the data by computing some basic statistics and graphical representations. This gives you a first insight into the data and facilitates the detection of possible errors, outliers, trends, patterns, etc. A scatterplot matrix provides a helpful overview of the data, but of course it is not practical when the number of variables is large. The following line shows it for the numerical variables in the air pollution data set (columns 3 to 9).

```r
pairs(airpol[, 3:9])
```

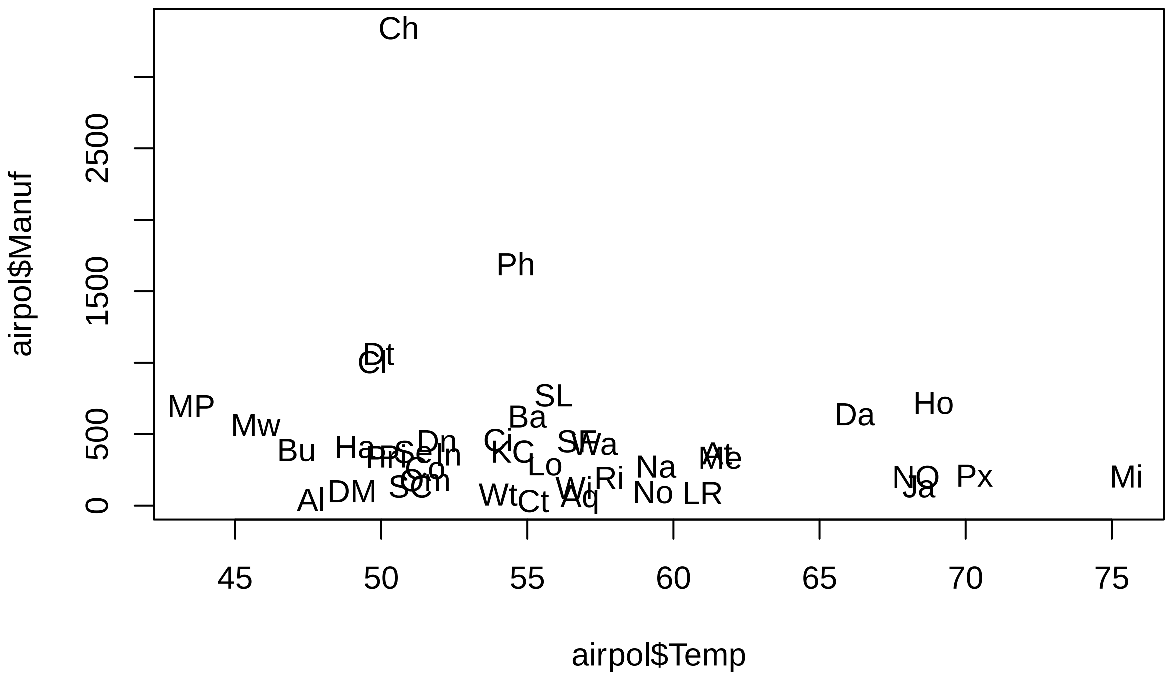


The single scatterplot of Temp against Manuf suggests an outlier (the State with the very large value for Manuf). However, the Pop by Manuf scatterplot, and several others, show that although this State is extreme it is not really an outlier. There are some clearly non-normally distributed variables. You can check this out by plotting some histograms using `hist` of qq-plots using `qqnorm`).

Labels to identify individual points in a scatter diagram are often an essential aid to interpretation.

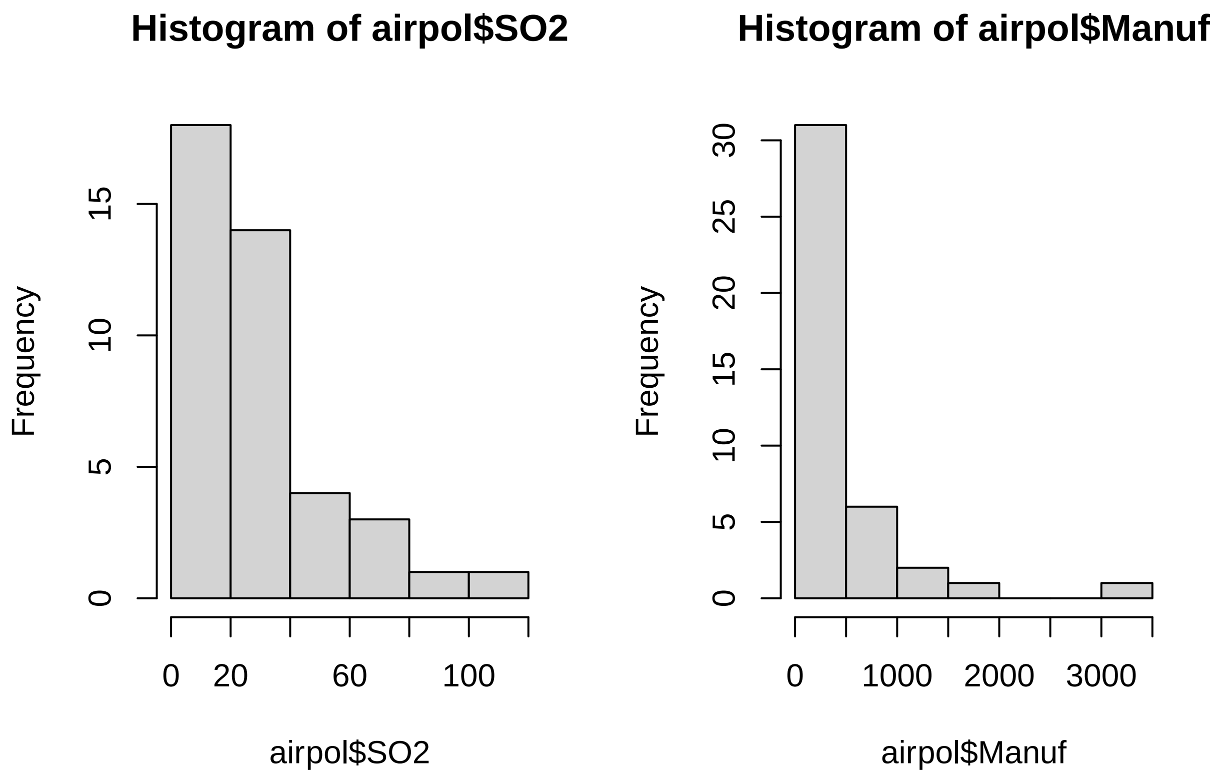
```r
# "n" -  do not plot the symbols
plot(airpol$Temp, airpol$Manuf, type = "n")
```

```r
# "text" adds labels to the current plot
text(airpol$Temp, airpol$Manuf, labels = airpol$Tn)
```



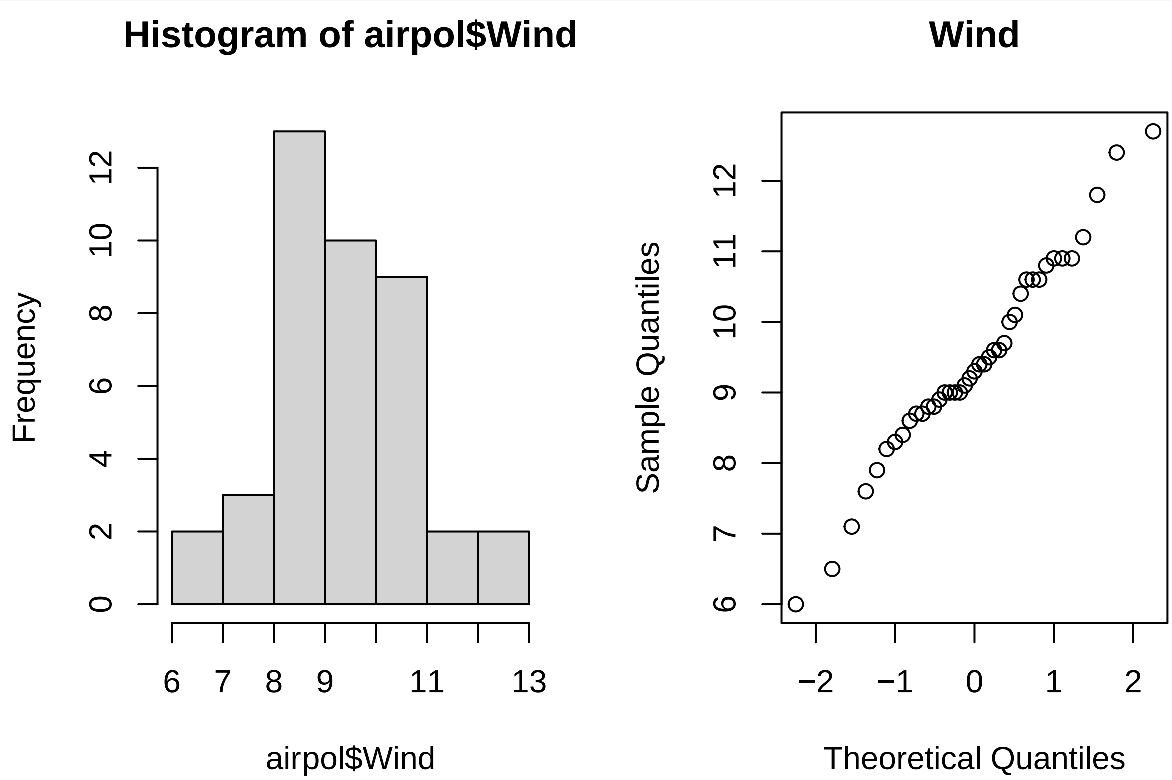
Chicago, a large city with a strong manufacturing base, was the potential outlier. The scatterplot matrix also reveals that there is one strong linear relationship, and possibly a second (Precip, Temp) though with a few States not fitting that pattern. Some non-normally distributed variables are also apparent.

```r
par(mfrow = c(1, 2)) # Show next graphs side-by-side
hist(airpol$SO2)
hist(airpol$Manuf)
```



Wind, despite being a discrete variable, appears approximately normally distributed.

```r
par(mfrow = c(1, 2))
hist(airpol$Wind)
qqnorm(airpol$Wind, main = "Wind")
```



As said, basic characteristics of the variables should be explored through summary statistics before proceeding to carry out any multivariate analysis. The most common being the sample mean and variance or standard deviation. The term *sample* refers to the fact that these measures are here computed from a real-world sample of data, in contrast with their *population* counterparts referring to the characteristics of the underlying random variables assumed to generate the observed data. When there is no possibility of confusion, they are simply called mean, variance, and so on. Given the ith variable (column) of the data set, the corresponding expressions are:

$$\begin{aligned}
\text{Sample Mean } \overline{x}_i &= \frac{1}{n}\sum_{k=1}^{n} x_{ki} \\
\text{Sample variance } s_i^2 &= Var(x_i) = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki}-\overline{x}_i)^2 \\
\text{Sample standard deviation } s_i &= \sqrt{s_i^2}
\end{aligned}$$

Basic summary statistics can be obtained in R as follows:

```
summary(airpol[, 3:9])
```

```
      SO2            Temp          Manuf           Pop            Wind      
 Min.   :  8   Min.   :43.5   Min.   :  35   Min.   :  71   Min.   : 6.00  
 1st Qu.: 13   1st Qu.:50.6   1st Qu.: 181   1st Qu.: 299   1st Qu.: 8.70  
 Median : 26   Median :54.6   Median : 347   Median : 515   Median : 9.30  
 Mean   : 30   Mean   :55.8   Mean   : 463   Mean   : 609   Mean   : 9.44  
 3rd Qu.: 35   3rd Qu.:59.3   3rd Qu.: 462   3rd Qu.: 717   3rd Qu.:10.60  
 Max.   :110   Max.   :75.5   Max.   :3344   Max.   :3369   Max.   :12.70  
     Precip          Days    
 Min.   : 7.05   Min.   : 36  
 1st Qu.:30.96   1st Qu.:103  
 Median :38.74   Median :115  
 Mean   :36.77   Mean   :114  
 3rd Qu.:43.11   3rd Qu.:128  
 Max.   :59.80   Max.   :166  
```

```
apply(airpol[, 3:9], 2, sd) # Compute standard deviations per variable
```

```
     SO2     Temp    Manuf      Pop     Wind   Precip     Days 
 23.4723   7.2277 563.4739 579.1130   1.4286  11.7715  26.5064 
```

We can notice that the scale and ranges of variation of the variables are very heterogeneous. With multivariate data we derive another fundamental summary statistic, the covariance matrix between every pair of variables.

$$\text{Sample covariance } s_{ij} = Cov(x_i, x_j) = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki}-\overline{x}_i)(x_{kj}-\overline{x}_j)$$

If the variable x_i is generally above its mean value whenever x_j is above its mean value then the covariance between them will be positive. If x_i is generally below its mean value whenever x_j is above its mean value then the covariance between them will be negative. The complete set of covariances for a set of variables is presented as a matrix.

```
options(digits = 2) # Sets number of digits to show in outputs
cov(airpol[3:9])
```

```
         SO2   Temp  Manuf    Pop   Wind  Precip   Days
SO2     550.9  -73.6   8528   6712   3.18   15.00  229.9
Temp    -73.6   52.2   -774   -262  -3.61   32.86  -82.4
Manuf  8527.7 -774.0 317503 311719 191.55 -215.02 1969.0
Pop    6712.0 -262.3 311719 335372 175.93 -178.05  646.0
Wind      3.2   -3.6    192    176   2.04   -0.22    6.2
Precip   15.0   32.9   -215   -178  -0.22  138.57  154.8
Days    229.9  -82.4   1969    646   6.21  154.79  702.6
```

Notice that this matrix is symmetric, that is $s_{ij} = s_{ji}$, and that the diagonal elements are the variances. Although covariance matrices play an important role in formal statistics, in practical data analysis they are not always easy to interpret and have the disadvantage that they are scale dependent. For example, multiply SO2 by 10 and all its covariances will increase by 10.

One of the most common scaling is to divide each variate by its own standard deviation. This forces every variate to have the same standard deviation, namely 1. In terms of variability all the variates are now on the same scale. A further consequence of this scaling is that the covariance matrix of the scaled variables is the so-called *correlation matrix* of the original variables.

$$\mathbf{R} = \begin{pmatrix} 1 & r_{12} & . & . & r_{1p} \\ r_{21} & 1 & r_{23} & . & r_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ r_{p1} & r_{p2} & r_{p3} & . & 1 \end{pmatrix},$$

where the pairwise correlations are computed from the covariances and standard deviations as $r_{ij} = \frac{s_{ij}}{s_i s_j}$. The correlation matrix plays a fundamental role in multivariate data analysis. Correlation coefficients measure the extent to which two variables are linearly related. For example, the correlation between Pop and Manuf is 0.955 and between Precip and Temp is 0.386.

```r
cor(airpol[3:9])
```

```
          SO2   Temp  Manuf    Pop   Wind Precip   Days
SO2     1.000 -0.434  0.645  0.494  0.095  0.054  0.370
Temp   -0.434  1.000 -0.190 -0.063 -0.350  0.386 -0.430
Manuf   0.645 -0.190  1.000  0.955  0.238 -0.032  0.132
Pop     0.494 -0.063  0.955  1.000  0.213 -0.026  0.042
Wind    0.095 -0.350  0.238  0.213  1.000 -0.013  0.164
Precip  0.054  0.386 -0.032 -0.026 -0.013  1.000  0.496
Days    0.370 -0.430  0.132  0.042  0.164  0.496  1.000
```

As several of the variables seem not be symmetrically distributed it is worth deriving the coefficient of skewness. This is estimated for a variable by the formula

$$g = \frac{1}{n} \sum_{k=1}^{n} (x_k - \overline{x})^3 \frac{1}{s^3}$$

The following code creates this formula in R:

```r
skew <- function(x) {
  m3 <- sum((x - mean(x)) ^ 3) / length(x)
  s3 <- sd(x) ^ 3
  m3 / s3
}
```

If applied to each numerical variable in the air pollution data we obtain

```r
apply(airpol[3:9], 2, skew)
```

```
    SO2    Temp   Manuf     Pop    Wind  Precip    Days 
 1.5841  0.8230  3.4846  2.9413  0.0027 -0.6925 -0.5501 
```

These values quantify what the histograms show. For example, Manuf is very positively skewed but Precip and Days have a slight negative skew (tail of the histogram to the left).

1.3 Multivariate normal distribution

Much of univariate statistical theory depends on the variable of interest having a normal probability distribution. We write $x \sim N(\mu, \sigma^2)$, where μ and σ^2 are the (population) mean and variance of the distribution of x. These two parameters define the distribution completely. Many univariate methods are *robust* to modest departures from normality which is why often we need assurance that our data are at least approximately normally distributed.

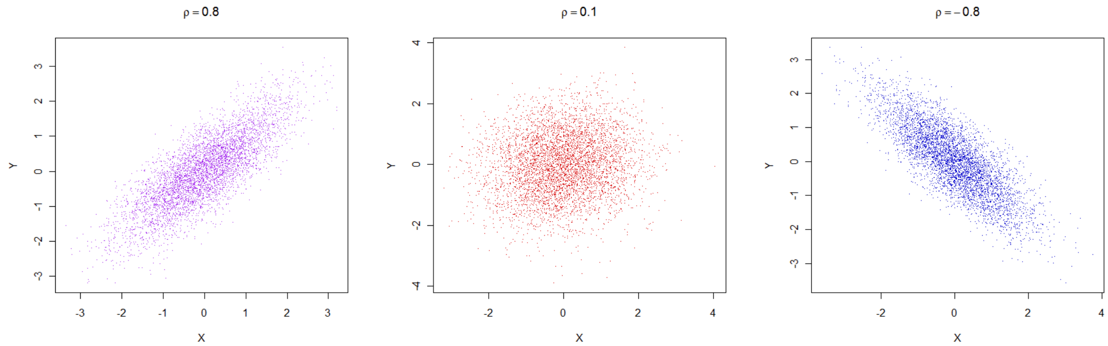
There is a set of multivariate statistical methods which similarly depend on the variables of interest jointly having a *multivariate normal distribution*. If we have p variables $x_1, \ldots, x_p$, we write these as a column vector $\mathbf{x}$ and say that $\mathbf{x} \sim N(\mu, \Sigma)$, where μ is the column vector of (population) means and Σ is the (population) covariance matrix.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}, \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{pmatrix}, \text{ and } \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & . & \sigma_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{p1} & \sigma_{p2} & \sigma_{p3} & . & \sigma_{pp} \end{pmatrix}$$

When $\mathbf{x}$ has a multivariate normal distribution, each of the individual x_i has a univariate normal distribution. The manner in which the x_i are related to each other is described by the covariance matrix. The covariance matrix Σ is of theoretical interest, however the (population) correlation matrix ρ can be conceptually more useful.

$$\begin{pmatrix} 1 & \rho_{12} & . & . & \rho_{1p} \\ \rho_{21} & 1 & \rho_{23} & . & \rho_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ \rho_{p1} & \rho_{p2} & \rho_{p3} & . & 1 \end{pmatrix}$$

Covariances and correlations are closely linked as shown above: $\rho ij = \sigma_{ij}/\sigma_i.\sigma_j$. In practice, correlations are more interpretable as they are unaffected by rescaling the variables, e.g. from cm. to mm., and the are defined in the interval $[-1, 1]$, with -1 meaning perfect negative correlation, 1 perfect positive correlation and 0 no correlation. The following figure shows exemplary cases of clouds of data points associated with correlations of 0.8, 0.1 and -0.8.



1.4 Exercises

1. NIR data. Use a scatterplot matrix to see whether there is any relationship between protein, moisture and the first 10 wavelengths. How would you describe the relationships among the wavelengths? Derive the correlation matrix for protein, moisture and the first 10 wavelengths and use it to strengthen your conclusions based on the plots only. Use boxplots to get an overview of all the wavelengths, how their mean values and variation change across the NIR spectrum, i.e. as you move from the first to the last wavelength.

2. SOUP data. Use a scatter plot matrix to get an overview of the data. Look for a relationship between Tomato Flavour and Tomato Odour. Are there other similar relationships? Derive the summary statistics for these variables, i.e. means, standard deviations, minimums and maximums. Also look at the correlation matrix for all the variables. It should confirm what you saw in the scatterplot matrix graph.

3. RIME ICE data. Derive the mean, standard deviation and coefficient of skewness for the 14 elements. Are any of these 14 elements correlated with each other? Generate the scatterplot matrix for the elements and confirm that the most skewed elements in the plots have the largest (positive or negative) skewness coefficients. Use histograms to see whether any element is approximately normally distributed. Are the means and variances of any of the elements affected by whether they face North or South? For this last question you will need R statements of the form `tapply(element,ns,mean)`.

2 Principal components analysis

2.1 Introduction

The central aim of principal component analysis (PCA) is reducing the size of a multivariate data set while still accounting for as much of the original variation as possible. In doing so it transforms a set of intercorrelated variables $x_1, \dots, x_p$ into a set of uncorrelated variables $y_1, \dots, y_p$, the so-called *principal components*. The idea is that if a few principal components are able to account for the main sources of variation in the original variables, without a commensurate loss of information, this will facilitate the analysis of the data set.

For example, car magazines, videogame magazines, and the like often compare a range of products on several criteria (variables) but additionally provide a single, overall, summary score. This summary score is meant to provide, loosely speaking, a weighted average of the scores on all the individual criteria.

With PCA we attempt something similar. We try to generate summary scores for the observations based on the values of all the individual variables. However, we often generate several summary scores from one set of variables and we allow more flexibility than just weighted averages. The summary variables in standard PCA are linear combinations of the original variables.

For example, suppose we wish to use the average of two responses, x and y. Clearly, we would use the linear combination

$$\frac{x+y}{2}$$

This can be written as $0.5x + 0.5y$. Another linear combination useful to compare x and y would be $x - y$, which can be also written $+1x - 1y$. When x is large, positive and y is large, negative, $x - y$ will be large, positive, the linear combination will have a large, positive value (a large, positive *score* in PCA). On the other hand when x is large negative and y is large positive, $x - y$ will have a large, negative score. When x and y are approximately equal $x - y$ will have a score close to zero.

Suppose we wish to compare x with the average of other two variables y and z. We could use

$$x - \frac{y+z}{2}$$

Again, we have a linear combination of the variables x, y and z and we can write this combination as $1x - 0.5y - 0.5z$. The coefficients 1, -0.5 and -0.5 are called the *loadings* in PCA. They indicate how much each variable contributes to the score and the manner (positive, negative) in which they contribute.

As another example, suppose we wish to compare the average of our first four variables x_1, x_2, x_3 and x_4 with the remaining ten variables $x_5, \dots, x_{14}$. A linear combination to achieve this can be

$$0.25(x_1 + x_2 + x_3 + x_4) - 0.1(x_5 + x_6 + \cdots + x_{14})$$

The 14 loadings take the values $+0.25$ or -0.1. When the $x_1, \dots, x_4$ are relatively large and the $x_5, \dots, x_{14}$ are relatively small, the score for this linear combination will be large, positive. When the $x_1, \dots, x_4$ are relatively small and the $x_5, \dots, x_{14}$ are relatively large, the score will be large, negative.

Hence, the objective of PCA is to produce optimal linear combinations (involving loadings and scores) which maximise the representation of the variation in the original data set using a few new summary variables. The principal components generate a simplified view useful for interpretation and plotting purposes:

- Sometimes they are useful to define a (smaller) set of new variables summarising the original ones. For example, to build a economic, environmental or ecological index; or to be able to manage and visualise hundreds or even thousands of variables as it is commonly the case with high-throughput data in the *omics* sciences.

- Other times they are just used as a technical device to reduce a high-dimensional data set where p is typically greater than n, e.g. to be able to conduct regression analysis with many explanatory and/or high correlated explanatory variables.

Some examples of real-world applications include:

- Building synthetic indicators (health, meteorological, environmental, economic indexes).
- Image analysis: identification of similar faces, phenotypes, image compression, ...
- Sensory analysis: visualise relationships among wine varieties and sensory attributes.
- Ecology: visualise relationships among several environmental variables and species.

2.2 Basic elements

Given a set of p variables $x_1, x_2, \ldots, x_p$ with covariance matrix Σ, each principal component (PC) y_i is a linear combination of the original variables:

$$y_i = l_{i1}x_1 + \ldots + l_{ip}x_p \quad i = 1, \ldots, p,$$

where the loadings $l_{i1}, \ldots, l_{ip}$ are computed in a way such that the variance of each PC y_i, $Var(y_i)$, is maximised and the PCs are uncorrelated with each other. As we can always find a linear combination giving rise to an infinite variance, a technical condition is imposed to determine a solution. In particular, the loadings are constrained so that $\sum_{k=1}^{p} l_{ik}^2 = 1$ for each PC y_i.

By construction, the PCs satisfy the following key properties:

- $\sum_{i=1}^{p} Var(y_i) = \sum_{i=1}^{p} Var(x_i)$, hence the total variability of the PCs always equals the total original variability.
- $Var(y_1) \geq Var(y_2) \geq \ldots \geq Var(y_p)$, hence the first PC explains most of the total variability, its variance $Var(y_1)$ has the maximum value. The variance of the second, $Var(y_2)$, is the maximum possible variance among all PCs after that of y_1. The variance of the third, $Var(y_3)$, is the maximum after those of y_1 and y_2, and so on.
- The covariance $Cov(y_i, y_j) = 0$ for any pair of PCs y_i and y_j.

Hence, every linear combination will maximise its respective variance, but they will be relative maxima to respect to each other so, under certain conditions, we will be able to arrange them in decreasing order. For a PC y_i, the ratio

$$\frac{Var(y_i)}{\sum_{k=1}^{p} Var(y_k)}$$

measures what fraction of the original variability is explained by the ith PC. If we consider the subset of the first q PCs (after ordering them according to the size of their variances),

$$\frac{Var(y_1) + \cdots + Var(y_q)}{\sum_{k=1}^{p} Var(y_k)}$$

this calculation informs about how much of the original variability is explained by the first q PCs. In many cases, a few PCs will account for most of the original variability. And so, the dimension of the multivariate data set can be reduced without losing too much information.

The loadings are useful to interpret the PCs. Their value is proportional to the correlation between the PCs and the original variables (as long as the original variances are similar). Once the loadings are obtained, the result of replacing the original values $x_1, x_2, \ldots, x_p$ in the PC equations provides the numerical values of the PCs $y_1, \ldots, y_p$, these are the scores.

2.3 PCA on standardised variables

PCA is not invariant to the scale of the variables. Often, when the original variables are all measured on unrelated scales, it makes sense to work with standardised variables. This us usually handled in practice by transforming the variables to have zero mean and unit variance. That is, instead of working with the variables x_i, we use

$$z_i = \frac{x_i - \overline{x}_i}{s_i}$$

The standardised variables z_i (a.k.a. z-scores) have all mean equal to 0 and variance/standard deviation equal to 1. Note that doing PCA on the $z's$ is equivalent to derive the PCs from the correlation matrix of the data set, instead of from the covariance matrix.

Without this adjustment the original variables with relatively large variances will have the largest loadings and will excessively influence the results, tending to dominate the first PCs and masking the relative contributions of the others. This is e.g. the case when mixing variables taking values in the thousands, like say annual earnings, with variables accounting for body weight in kilograms. In order to give all the variables the same relevance in the solution it is necessary to standardise.

However, note that the PCAs conducted on the original scale of the variables x_i and the standardised scale of the z_i will generally lead to different results with no simple correspondence between them. Thus, to work with original or standardised variables must be a thoughtful decision in the context of the application and accounting for the characteristics of the variables. Note that some software packages and researchers standardise the variables by default. However we recommend that this must be assessed on a case by case basis. For example, when working with spectral data typically consisting of many variables, standardising will have the (generally undesirable) effect of giving the noise signals the same relevance as the signals presenting genuine biological processes.

Finally note that when PCA is based on the correlation matrix (standardised variables), then the loadings do are exactly proportional to the correlation between the PCs and the original variables.

2.4 PCA of the air pollution data set

The data were collected as part of an air pollution study across cities in the USA (Everitt & Hothorn, 2011). The following variables were obtained for 41 US towns. An objective of the study was to investigate how the pollution level (SO_2) was explained by the six other environmental variables, which sounds like a regression-type analysis. We will focus here on using PCA to explore the relationships between those six environmental variables and characterise the towns according to them. The R file with the data set is called `air_pollution.Rdata`.

2.4.1 Basic R functions involved

`princomp`: performs PCA on the covariance or correlation matrix using a a mathematical method called spectral decomposition. Check `?princomp` for details and options.

`prcomp`: performs PCA on the data matrix using a method called singular value decomposition. Check `?prcomp` for details and options. In general, this is regarded as more numerically accurate than PCA based on the spectral decomposition (`princomp`). This might be relevant for some large scale applications, but no different will be generally found in standard applications.

`biplot`: works on two sets of two-dimensional points providing the coordinates for the biplot representation. Alternatively, it can take `princomp` and `prcomp` class objects as input.

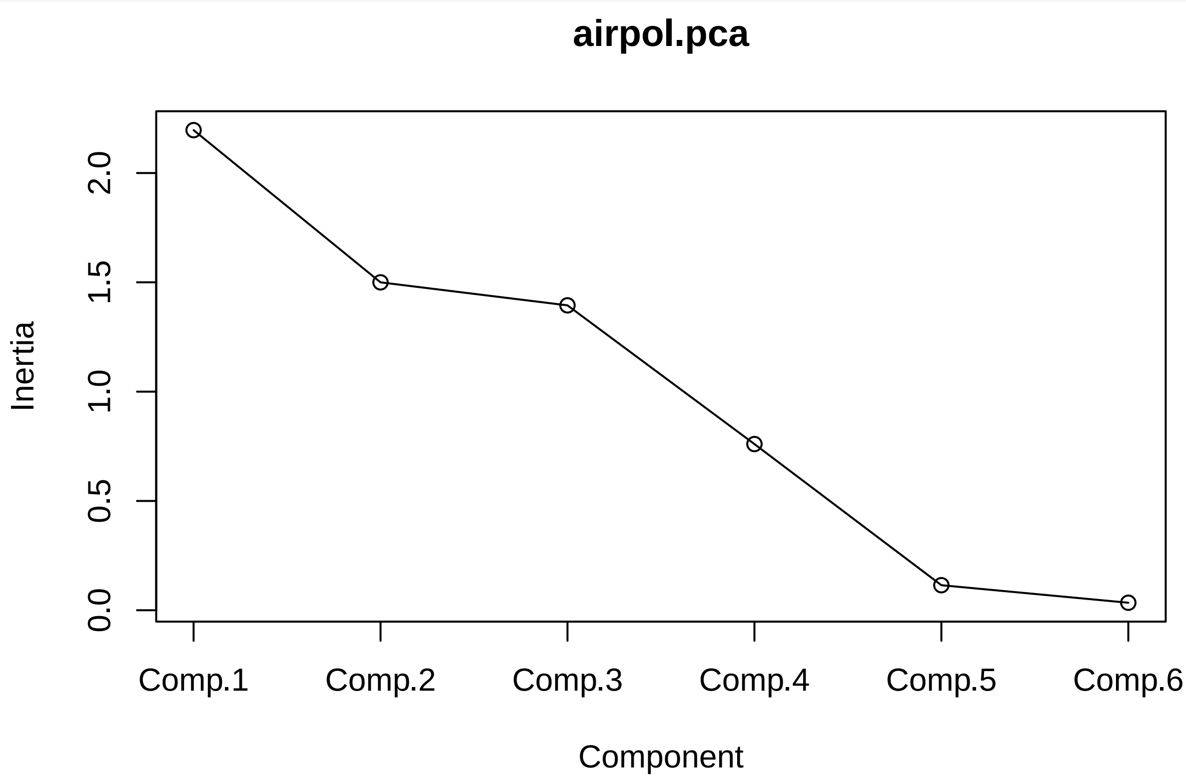
We begin by deriving the PCs of the air pollution data, excluding SO2 (this one is considered in this setting a variable which is more a response to the environmental conditions represented by the others). Because the variables are on fairly heterogeneous and unrelated scales we standardise them; that is, we will do PCA based on their correlation matrix (check out `?princomp` for details).

```
# Load the data (if not already loaded)
airpol <- read.table("air_pollution.txt", h = T, stringsAsFactors = TRUE)
# Correlation-based PCA on the environmental variables (columns 4 to 9)
airpol.pca <- princomp(airpol[, 4:9], scores = T, cor = T)
options(digits = 3) # Control number of digits in results
summary(airpol.pca)
```

```
Importance of components:
                       Comp.1 Comp.2 Comp.3 Comp.4 Comp.5  Comp.6
Standard deviation      1.482  1.225  1.181  0.872 0.3385 0.18560
Proportion of Variance  0.366  0.250  0.232  0.127 0.0191 0.00574
Cumulative Proportion   0.366  0.616  0.848  0.975 0.9943 1.00000
```

The first component accounts for 36.6% of all the variation among the 6 variables. The first 3 PCs account for 85% of all the variation. It might be sufficient to proceed with only these 3 new variables rather than all 6 of the original variables. Using the first 2 PCs is in practice convenient for graphical representations. This information can be indirectly displayed by a scree plot.

```
screeplot(airpol.pca, type = "l")
```



This graph shows the variance explained by each PC. An "elbow" shape indicates from where considering more PCs does not imply a large gain in explained variance. Of course if all PCs as used then 100% of the original variability is explained, but no reduction in dimension is attained. Hence, this is the practical trade-off between aiming to explain as much as possible without using too many PCs. Typically explained variability above ~75% is considered good, but this depends on the context of application (e.g. with high-throughput data including several tens or hundreds of variables the first PCs often explain relatively low fractions). However, it is not uncommon that a few PCs accounting for a relatively low fraction of variability represent well the main sources of variation in the data set. In our case study, using 2-3 PCs as discussed before seems to be a sensible choice.

It can be convenient to extract the scores (these are the y values above) for each observation and the PCA loadings (the l coefficients above) to self-contained objects to be used later.

```
scores <- airpol.pca$scores
# Label rows with town names for easy reference
row.names(scores) <- airpol$Tn
# Show scores in first 3 PCs for the first 10 towns
scores[1:10, 1:3]
```

```
   Comp.1  Comp.2 Comp.3
Px 2.4401  4.1911  0.942
LR 1.6116 -0.3425  0.840
SF 0.5021  2.2553 -0.227
Dn 0.2074  1.9632 -1.266
Ha 0.2191 -0.9763 -0.595
Wt 0.9961 -0.5007 -0.433
Wa 0.0229  0.0546  0.354
Ja 1.2278 -0.8491  1.876
Mi 1.5332 -1.4047  2.607
At 0.5990 -0.5872  0.995
```

```
loadings <- airpol.pca$loadings
loadings[, 1:3] # Show loadings
```

```
        Comp.1 Comp.2  Comp.3
Temp    0.3296  0.128  0.6717
Manuf  -0.6115  0.168  0.2729
Pop    -0.5778  0.222  0.3504
Wind   -0.3538 -0.131 -0.2973
Precip  0.0408 -0.623  0.5046
Days   -0.2379 -0.708 -0.0931
```

2.4.2 Interpreting the PC loadings

The following points summarise general guidelines for interpreting the loadings:

- When all loadings are very similar in size and sign, then the PC is kind of an average of all variables. It is e.g. the case of a PC representing an overall economic or ecological index summarising the information across several variables.
- When loadings are very similar but some positive, some negative, then the PC represent a contrast, a comparison between variables in the data set.
- If some loadings are approximately zero, then the corresponding variables are not associated with the PC.
- Sometimes interpretation is not easy or possible, particularly for later components.

For the air pollution data, PC1 is a comparison between Temp and {Manuf, Pop, Wind, Days}. Towns with above average temperature and below average for all of {Manuf, Pop, Wind, Days} will have a large positive score. Towns with below average temperature and above average for all of {Manuf, Pop, Wind, Days} will have a large negative score. We interpret PC1 as 'environment' or 'quality of life'. Towns with large positive scores are warm, dry, calm, low population and without too many manufacturing businesses. Towns with a large negative score are cool, wet, windy, busy and industrial.

PC2 is dominated by rain. Towns with a large positive score are frequently very wet; towns with a large negative score are usually dry and do not experience very wet weather.

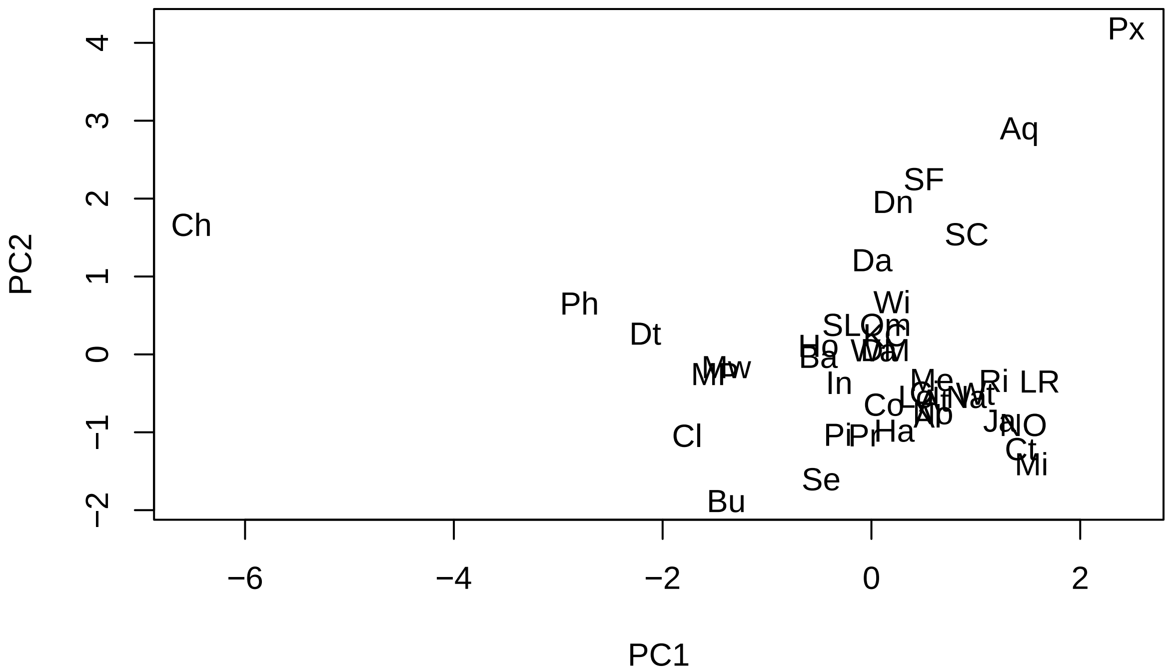
PC3 defies interpretation and that is not uncommon. Principal components are derived according to the statistical criterion of maximising variation and ignoring any underlying biological or other structure in the data.

Note that there is indeterminacy in the loadings. For any PC all the loadings can be multiplied by -1 without causing any fundamental change. The % variance remains the same. The order of the scores is simply reversed, but the differences in absolute values and interpretations of the PCs remain the same. Note also that the second table of results shown by `princomp$loadings` (SS loadings and so on) is actually fairly useless in practice.

2.4.3 Plotting the PC scores

The PC scores are new variables representing the observations. The can be used to display them in a scatterplot. The following figures shows this using the first two PCs of the air pollution data set. The town labels stored in the variable `Tn` are used to identify them.

```
plot(scores[, 1], scores[, 2], type = "n", xlab = "PC1", ylab = "PC2")
text(scores[,1], scores[,2], airpol$Tn)
```



We see in this plot that Chicago (Ch) is the large, busy, slightly windy and slightly wet city and that Phoenix is a small, quiet, very calm, very dry city (Px). Be aware that because of the indeterminacy of PCA sometimes different software packages can appear to produce different solutions, or even when running PCA several times on the same system. But it is simply that the loadings have been reversed from one to another for one or several PCs. The scores will appeared reversed in the graph, but the relative positions of the points will be exactly the same.

2.4.4 Relating the PCs to SO2

We can see whether these new PC variables are linked to Sulphur Dioxide computing ordinary correlations. We use the `round` function to obtain the output in a more readable format.

```
round(cor(cbind(airpol$SO2, scores[, 1:3])), digits = 2)
```

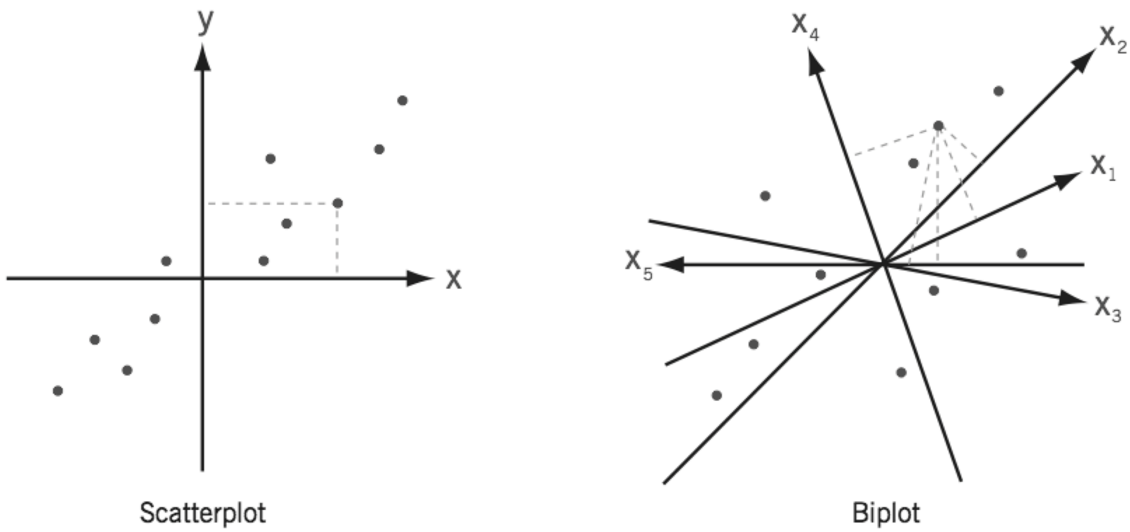
```
              Comp.1 Comp.2 Comp.3
         1.00  -0.64  -0.12   0.02
Comp.1 -0.64    1.00   0.00   0.00
```

```
Comp.2 -0.12   0.00   1.00   0.00
Comp.3  0.02   0.00   0.00   1.00
```

Clearly PC1 has a moderate negative correlation with SO2 but none of the other PCs have. Note, in passing, the zero correlation among the PCs as expected.

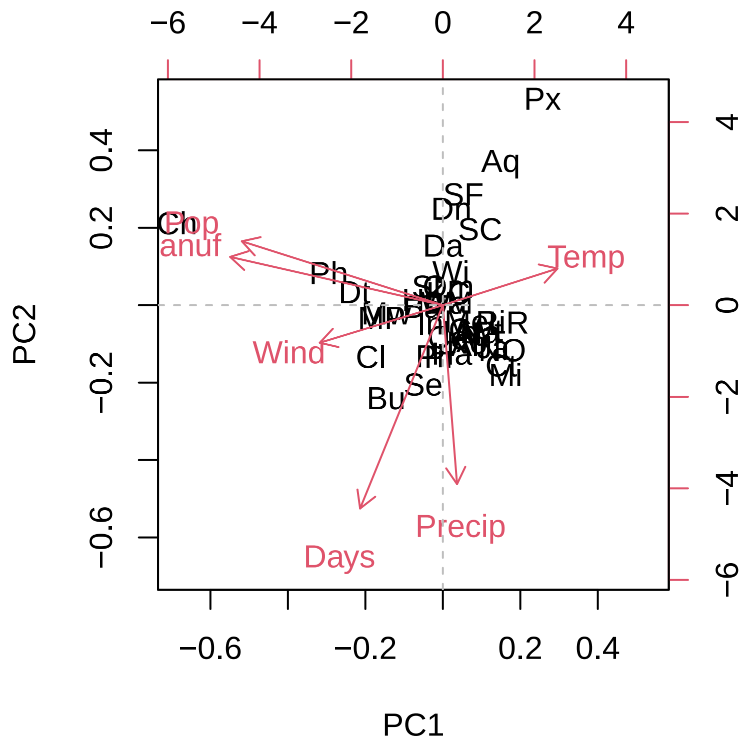
2.5 PCA biplots

A biplot can be understood as a generalisation of the scatterplot of two variables to the case of many variables. While a simple scatterplot has two perpendicular axes, biplots have as many axes as there are variables. However, in a biplot the values are represented only approximately in a space of lower dimensionality, usually 2-dimensional using the first two PCs.



Biplots provide a very useful summary of the results of a principal component analysis (note that they can be also produced from other multivariate techniques). The following code produces the PCA biplot for the air pollution data set using the first two PCs (check out `?biplot`). Reference lines are useful to facilitate interpretation.

```
biplot(airpol.pca, xlabs = airpol$Tn, choices = c(1, 2), xlab = "PC1", ylab = "PC2")
# Add reference lines
abline(h = 0, lty = 2, col = "grey")
abline(v = 0, lty = 2, col = "grey")
```



In the biplot below both variables and observations are jointly displayed. The segments from the origin (commonly called rays or axis) represent the original variables and the points are the scores on the first two PCs (note that the configuration of the points is the same as in the scatterplot of the scores we produced manually above). Occasionally a biplot of components other than the first two is helpful. The value of the biplot as an approximation to the entire data set is only as good as the extent to which the first two PCs explain all the variation in the data, in this case 62%.

Main points to interpret the biplot display are the following:

- The angle between rays indicates correlation between the corresponding variables.
  - Rays in the same direction indicates positive correlation (rays approx. overlapping indicates correlation approx. 1).
  - A 90 degrees angle between them indicates no correlation.
  - Rays in opposite directions indicate negative correlation (approx. straight angle indicates correlation approx. -1).
- The length of a ray approximates the standard deviation of the variable (if not standardised). If the variables are standardised to have unit variances, then the lengths indicate how well variables are represented.
- The angle of the rays with the PC reference lines determined by the loadings and indicates how the variables relate with each PC.
- The distance between points approximates the similarity between observations in terms of the original variables.
- Observations (points, PCA scores) can be ranked on each original variable by their orthogonal projection onto the (extended) variable axis (rays). For example, extend the line for POP in both directions. Now drop a perpendicular from say, Ch, onto this extended line. The distance from where the perpendicular hits the Pop ray to the origin measures the (relative) value of Ch for Pop. Measuring the value of all other towns in the same way shows immediately the rank order of the cities with respect to Pop. It also shows which cities are above the mean value of Pop: those on the positive side of the ray/axis. And it shows those below the average: those on the negative side of the ray, the side which has to be extended backwards from the origin.

- The % variance explained by selected PCs (usually the 1^{st} and 2^{nd} PCs) measures extent to which the biplot summarises the entire data set.

The biplot of the air pollution data set actually shows all the main features discussed before.

Some highlights of the analysis by columns (variables):

- Manuf and Pop show the highest correlation, both weakly correlated with Days and Precip.
- Strong negative correlation between Temp and Wind according to the angle between their rays, although they are not very well represented in the biplot (short ray lengths in a correlation biplot, check correlation matrix for actual correlation between them).
- The horizontal axis (PC1) mostly accounts for a contrast between Temp and {Manuf, Pop, Wind} (the corresponding rays with origin at (0, 0) show the smallest angles in relation to this axis). This is in accordance with the analysis of PCA loadings above. This allows to discuss the relative environmental characteristics of the cities according to these two groups of variables on the basis of their values in PC1.
- The vertical axis (PC2) is mostly associated to Precip and Days. Thus, cities with negative values in PC2 will be more rainy than those with positive values.

Some highlights of the analysis by rows (towns):

- Chicago, a large city with a strong manufacturing base, was the potential outlier. Also, Phoenix seems to have a particular profile.
- No obvious clusters of cities based on the environmental variables considered.

Some highlights of the joint analysis of rows and columns (perpendicular projection of points onto the biplot (extended) axes/rays):

- Chicago distinguished by very high levels over the average in Manuf and Pop.
- Wind-Temp define an axis ordering cities in relation to wind and temperature, from Chicago (the windiest) and cities like Philadelphia, Detroit, Cleveland or Buffalo on the left-hand side (temperature below average, wind over average) to cities like Albuquerque and Phoenix (the warmest) on the right-hand side (temperature over average, wind below average).

2.6 Exercises

1. RIME ICE data. Derive the principal components of the Rime Ice data, deciding first whether to standardise the elements (i.e. whether to use variance-covariance or correlation PCs). Do the PCs provide a good summary of all the variation among the elements? How would you interpret the first two components? Generate a biplot and make sure you can interpret it.
2. Air Pollution data. Does taking logs of the positively skewed variables improve the PC analysis, i.e. does it cause the % variance accounted for to increase for the first few PCs? Are the loadings of the first two PCs amenable to the same interpretation as they were before transformation?
3. NIR data. Which principal components are correlated with protein and which with moisture? Generate a scatter plot matrix for protein, moisture and the scores on the first 6 components. Why is a biplot not so useful with these data? Note that using `princomp` does not work here. This is because there are more variables, 68, than samples, 39. The alternative function `prcomp` does work in this case using a different technique to produce the PCA solution that does not rely explicitly on the covariance or correlation matrix. However the output of `prcomp` is somewhat less user friendly.

```
nir <- read.table("nir.txt", h=T, stringsAsFactors = TRUE)
nir.pca <- prcomp(nir[,3:70], retx=T) #return the scores in 'x'
# Loadings for the first 5 variables (wv) in the first 6 PCs
nir.pca$rotation[1:5,1:6]
```

```
          PC1     PC2   PC3    PC4    PC5    PC6
wv1 0.00863 0.02203 0.168 0.0669 -0.213 0.2062
wv2 0.01631 0.00530 0.164 0.0839 -0.170 0.1238
wv3 0.02411 0.00668 0.158 0.0812 -0.142 0.0952
wv4 0.02883 0.01934 0.159 0.0759 -0.114 0.1084
wv5 0.02802 0.02122 0.164 0.0303 -0.110 0.1162
```

```r
# Scores in the first 6 PCs
scores <- nir.pca$x[,1:6]
# Var. explained by the first six PCs
summary(nir.pca)$importance[,1:6]
```

```
                         PC1    PC2     PC3     PC4     PC5     PC6
Standard deviation     0.183 0.0184 0.00863 0.00628 0.00421 0.00183
Proportion of Variance 0.986 0.0099 0.00218 0.00116 0.00052 0.00010
Cumulative Proportion  0.986 0.9959 0.99806 0.99922 0.99974 0.99984
```

```r
# Correlation between response variables and first 3 PCs obtained from the wv variables
cor(nir[,1:2],scores[,1:6])
```

```
           PC1     PC2     PC3     PC4     PC5      PC6
protein  0.699  0.0925 -0.1891  0.6677 -0.1079 -0.00948
moisture 0.163 -0.9735 -0.0908 -0.0444 -0.0625 -0.02163
```

3 Discriminant analysis

3.1 Introduction

As discussed in the previous section, principal component analysis is about describing the major sources of variation by a small number of new variables. Discriminant analysis uses a similar strategy but with the aim of finding new variables (so called canonical variates or discriminant scores) which best discriminate among known groups in the data.

These new variables are also linear combinations of the original variables but now, if we plot the observations on the new axes, we will see as clear a separation among the groups as is possible.

In the data spreadsheet, in addition to a numerical data matrix $\mathbf{X}$, describing features of the samples, there will be a factor variable indicating which **group** or category each sample belongs to. For example, we might have samples from each of three species (groups) of flowers, and the species of each sample is recorded in a factor variable. Four features were measured from each sample (length and width of sepal and petal) and these are the numerical features of the samples.

Common questions that can be approached by discriminant analysis are:

- Can we allocate new samples to their correct (unknown) group? (prediction/classification goal)
- Which variables are involved in separation of the groups? (explanatory goal)

Some examples of application include:

- Plants belonging to different species: observations may be dimensions, colours, etc.
- Patients with different disease types: observations are signs and symptoms
- Foetuses with and without Down's syndrome: observations may be the mother's age and the concentrations of *human chorionic gonadotrophin* and *pregnancy-associated plasma protein* in her blood
- In archaeology, the sources of stone, clay or ore from which artefacts were made: observations may be concentrations of trace elements or of isotopes
- Applicants for credit who would and would not default.

Discriminant analysis broadly speaking belongs to a family of methods having data-driven classification and prediction into pre-determined classes as the main objective, what is also known as *supervised learning* in the machine learning arena.

3.2 Basic elements

Given p variables $x_1, \dots, x_p$, and a grouping factor (g groups), we re-describe them in terms of new variables y_i such that

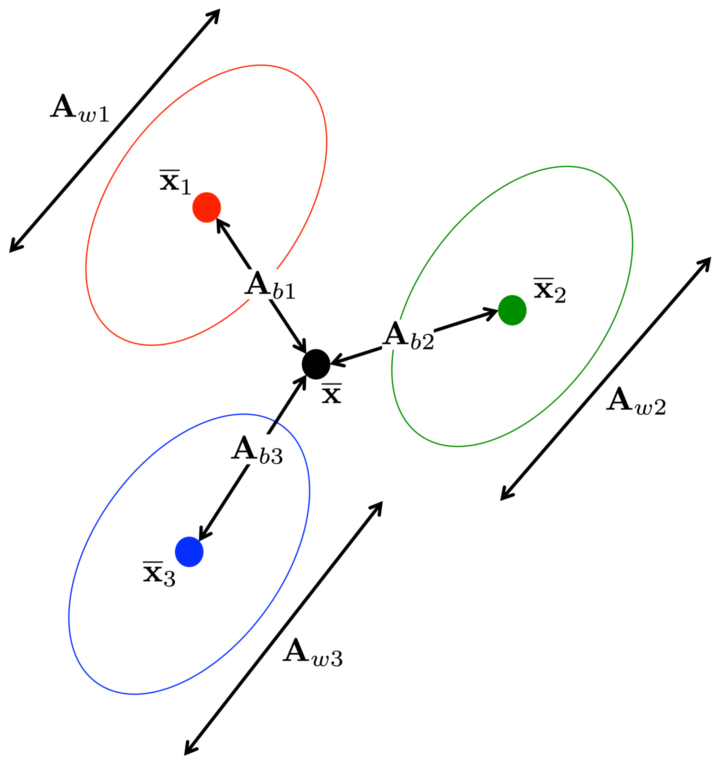
$$y_i = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{ip}x_p, \quad i = 1, \dots, g-1 \ (g < p)$$

The a_{ij} play the same role as the loadings in PCA. However they are determined in a different way so that the a_{ij} allow the y_i to separate groups as best as possible.

Fisher's Criterion:

$$\max \frac{\text{between-group variability}(\mathbf{A}_b)}{\text{within-group variability}(\mathbf{A}_w)}$$

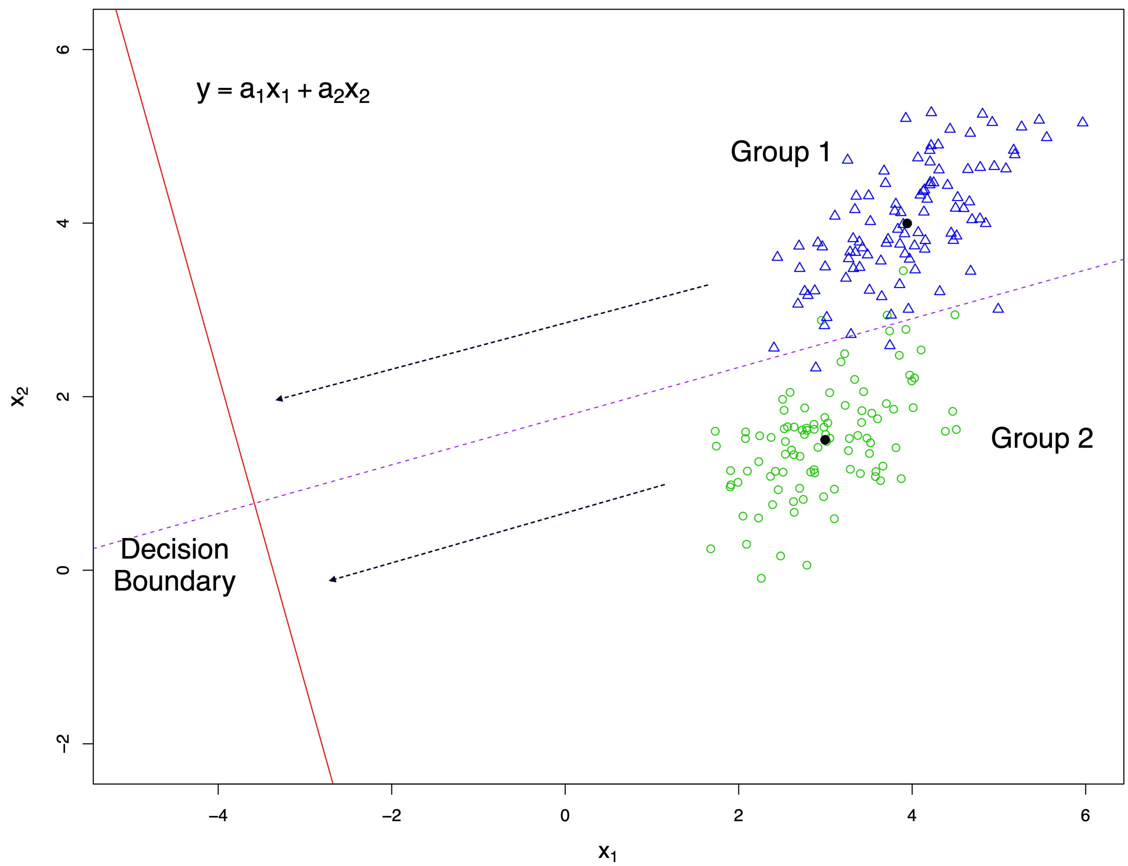
The following figure illustrates these different sources of variability in the case of three groups.



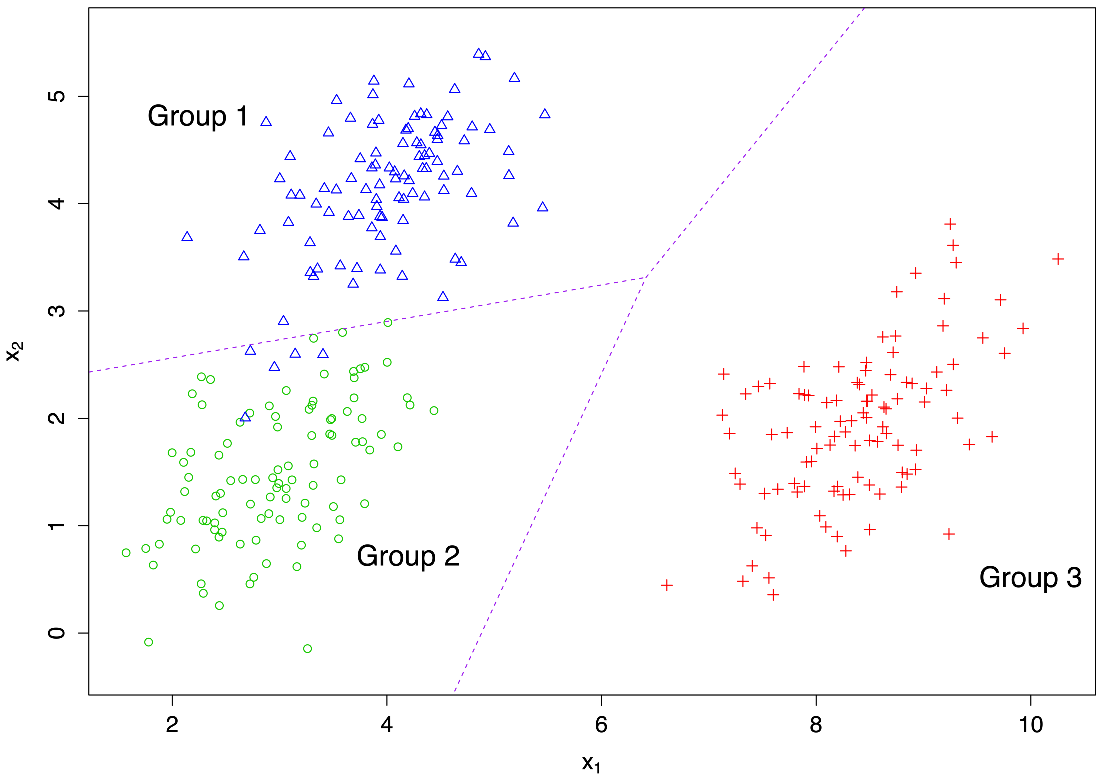
The new **uncorrelated** variables $y_1, \dots, y_{g-1}$ have by construction decreasing discrimination power. Normality and homogeneous group covariance matrices are typically assumed for $x_1, \dots, x_p$.

Discriminant analysis provides an optimum solution in the sense that it finds the linear combination of the original variables which maximises the ratio of between-group variation to within-group variation. The maximum number of canonical variates which can be derived is the smaller of $(g-1)$ and p. The scores from the canonical variates are uncorrelated with each other as in PCA.

The following figure illustrates graphically the general idea of discriminant analysis for the case of two variables (x_1, x_2) and two groups. Points and group centres are projected onto the discriminant score axis (red solid line). Samples are allocated to a group according to their closeness to the projected group centres. A threshold (decision boundary) is computed to determine group allocation.



The following figure shows another example for the case of two variables x_1 and x_2 and three groups.



3.3 Discriminant analysis of the pea pods data set

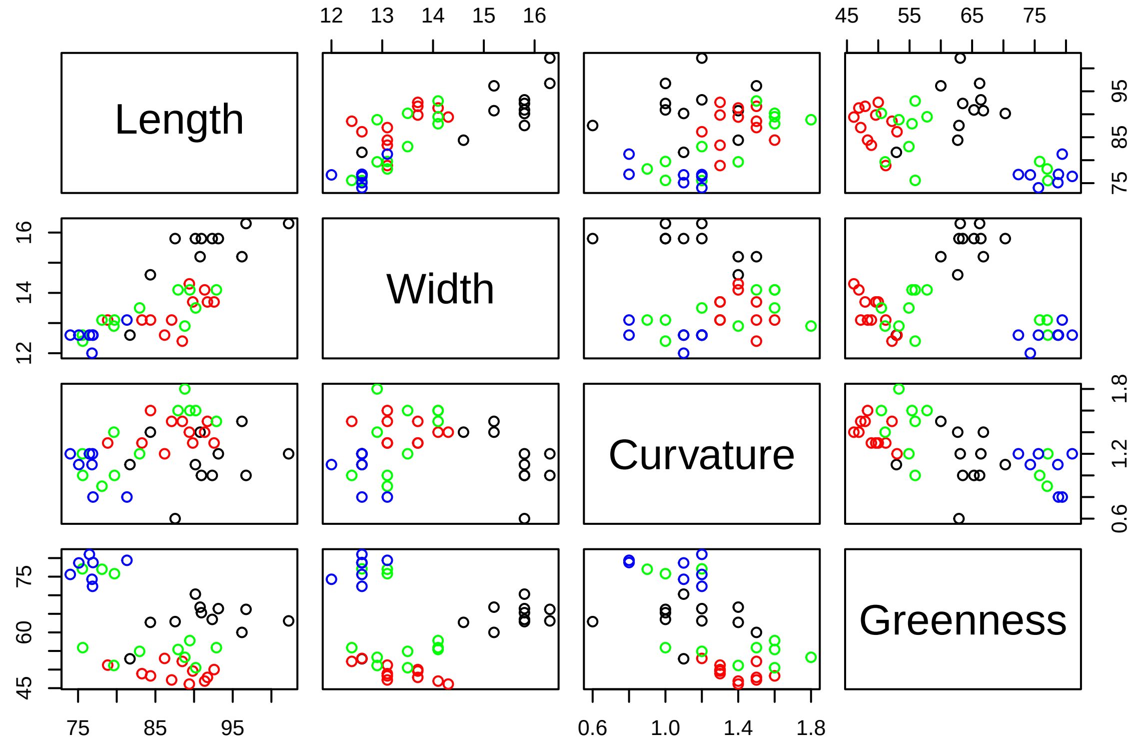
The data consist of 40 pea pods: 11 each of varieties A, B, C and 7 of variety D were collected, and the length(mm), width(mm), curvature(mm) and greenness (intensity units) of each pod was recorded. Curvature is the minimum distance from an end-to-end straight line to the top edge of the pod. These variables will be used as *classifiers* or *predictors*.

```r
pods <- read.table("pea_pods.txt", h = T, stringsAsFactors = TRUE)
str(pods)
```

```
'data.frame':   40 obs. of  5 variables:
 $ Variety  : Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 1 1 1 1 1 ...
 $ Length   : num  84.3 91 87.5 96.7 90.8 ...
 $ Width    : num  14.6 15.8 15.8 16.3 15.2 15.8 15.8 15.8 15.2 16.3 ...
 $ Curvature: num  1.4 1 0.6 1 1.4 1 1.1 1.2 1.5 1.2 ...
 $ Greenness: num  62.7 65.3 62.9 66.2 66.8 63.5 70.3 66.4 60 63.1 ...
```

We want to know if we can use our 4 variables to discriminate cleanly among the four varieties A, B, C and D. Clearly, from the graph below there is some scope to achieve this as the varieties can be distinguished based on some pairs of predictors, but it is not clear how best to proceed.

```r
# Define colours for varieties A, B, C and D respectively
colours <- c("black", "red", "green", "blue")[pods$Variety]
# Show data in scatterplot matrix
pairs(pods[, 2:5], col = colours)
```



Linear discriminant analysis (LDA) is implemented in the `lda` function of the `MASS` package.

```
library(MASS)
pods.lda <- lda(Variety ~ Length + Width + Greenness + Curvature, data = pods)
pods.lda
```

```
Call:
lda(Variety ~ Length + Width + Greenness + Curvature, data = pods)

Prior probabilities of groups:
    A     B     C     D 
0.275 0.275 0.275 0.175 

Group means:
  Length Width Greenness Curvature
A   91.5  15.4      63.6      1.14
B   87.6  13.4      49.2      1.39
C   83.7  13.3      60.4      1.35
D   76.8  12.6      77.2      1.06

Coefficients of linear discriminants:
               LD1     LD2     LD3
Length     0.00165  0.0600 -0.2042
Width     -0.96949 -1.2496  1.0505
Greenness  0.13689 -0.0909  0.0112
Curvature  1.19286 -0.0742  5.1818

Proportion of trace:
  LD1   LD2   LD3 
0.564 0.421 0.015 
```

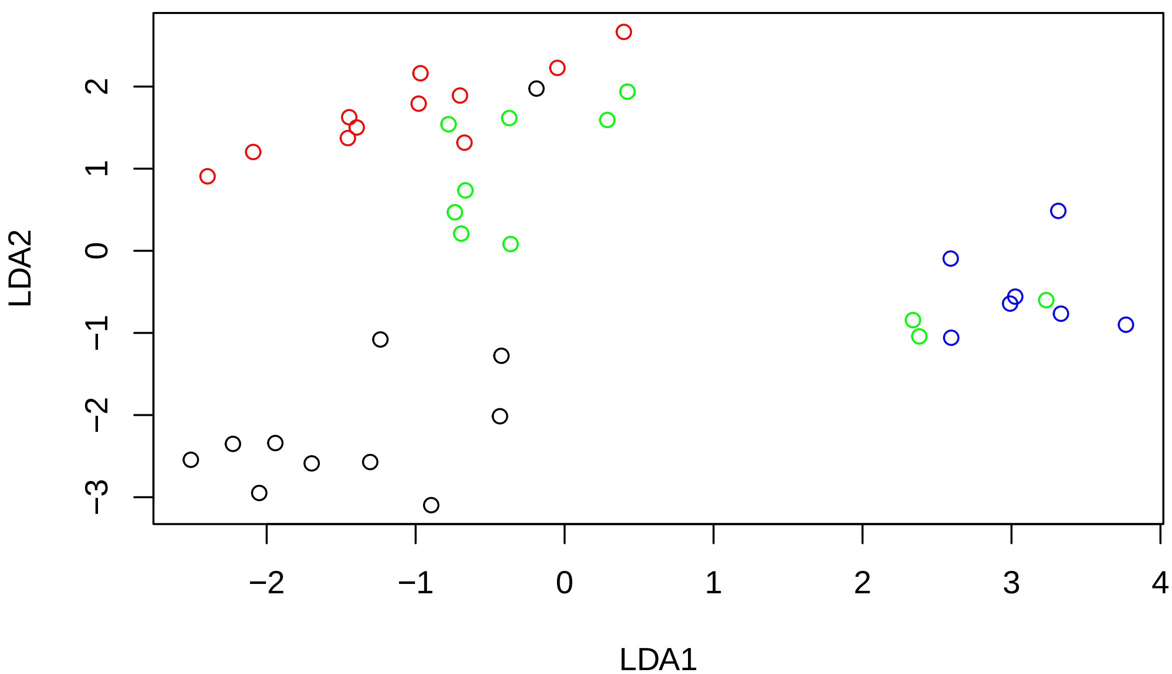
The output from the function `lda` provides the group means, the "loadings" a_{ij} of the canonical variates (coefficients of linear discriminants), and their % discriminatory power (proportion of trace). The scores and other elements are also stored in the `lda` object. The first canonical variate contrasts Curvature with Width, while the second is essentially representing Width alone. In turn these account for 56.44% and 42.06% of the discriminatory power (hence, reaching 96% of the between group variability explained in total). The interpretation of the coefficients is conducted as for PCA (note also that the indeterminacy property holds here, so you might find instances were the coefficients appear multiplied by -1). One conclusion from this analysis might be that pod Length and pod Greenness do not vary according to Variety, at least not sufficiently to assist in Varietal identification.

The canonical variate loadings are generated directly from the function `lda` but the remaining useful information is obtained from the function `predict`, including the scores as shown above. This generates for each sample three results:

- $class: predicted group allocation.
- $posterior: the estimated probability of belonging to each group.
- $x: sample scores on each canonical variate.

The discriminant scores can be used to represent the observations in low dimensions as in the PCA biplot and explore the overall similarity between groups. The following code extract the scores for the first two canonical variates (LDA1 and LDA2) and plot them.

```
pods.scores <- predict(pods.lda, dimen = 2, method = "plug-in")
plot(pods.scores$x[, 1], pods.scores$x[, 2], col = colours, xlab = "LDA1", ylab = "LDA2")
```



Variety D is the most distant from the others along the first canonical variate (LDA1), i.e. in terms of the contrast between Curvature and Width. It is interesting to observe that 3 samples of variety C (green) appear far from the others and very much mixed with the variety D samples. This is suggesting an error when registering the variety of those three samples. Variety A appears to be the most clearly distinguished as it is the least mixed with the others.

Note the section of the output referring to prior probabilities of groups. LDA uses these weights in its derivation of the canonical variates. The default weights are the relative frequencies of the factor levels, Variety in our case. This will be reasonable only if these frequencies reflect the true (usually unknown) relative frequencies of the levels of the factor we are trying to predict. However, when allocating a new sample to a group it is generally assumed that the new sample is equally likely to belong to each of the groups. Some discriminant analyses are very sensitive to the prior probabilities and you should always investigate the consequences of not using equal prior probabilities. If the groups are severely imbalanced, the performance is generally very biased against the smallest group.

With the pods data, the lower frequency of variety D has no implications for the likely variety of an unknown pod so we will change to equal prior probabilities.

```
pods.lda <- lda(Variety ~ Length + Width + Greenness + Curvature,
                data = pods, prior = c(.25, .25, .25, .25))
pods.lda
```

```
Call:
lda(Variety ~ Length + Width + Greenness + Curvature, data = pods,
    prior = c(0.25, 0.25, 0.25, 0.25))

Prior probabilities of groups:
   A    B    C    D 
0.25 0.25 0.25 0.25 

Group means:
  Length Width Greenness Curvature
A   91.5  15.4      63.6      1.14
B   87.6  13.4      49.2      1.39
C   83.7  13.3      60.4      1.35
D   76.8  12.6      77.2      1.06

Coefficients of linear discriminants:
                LD1     LD2     LD3
Length     0.000132 -0.0594  0.2044
Width     -0.869075  1.3324 -1.0365
Greenness  0.144546  0.0776 -0.0143
Curvature  1.090054 -0.0510 -5.2047

Proportion of trace:
   LD1    LD2    LD3 
0.6340 0.3528 0.0132 
```

Equal prior probabilities have not changed the loadings very much, but the first canonical variate now has increased discriminatory power.

3.4 Prediction performance

The table below, occasionally called the confusion table, shows the classification performance of the canonical variates (using the first LDA model fit considering unequal prior probabilities). The rows show the "true" classifications, the columns show the predicted classifications. With perfect prediction all the off-diagonal elements of the table would be zero.

```
addmargins(table(pods$Variety, pods.scores$class))
```

```
      A  B  C  D Sum
  A  10  1  0  0  11
  B   0 11  0  0  11
  C   0  3  5  3  11
  D   0  0  0  7   7
  Sum 10 15  5 10  40
```

The percentage of correctly classified samples provides the overall accuracy of 82.5%. Interpretation can be easier by using proportions (by rows in the table below).

```
prop.table(table(pods$Variety, pods.scores$class), 1)
```

```
         A      B      C      D
  A 0.9091 0.0909 0.0000 0.0000
  B 0.0000 1.0000 0.0000 0.0000
  C 0.0000 0.2727 0.4545 0.2727
  D 0.0000 0.0000 0.0000 1.0000
```

The predictions and performance above use the fitted model to classify the original samples into their known groups. This is helpful to have a crude assessment of prediction performance. This is the default *re-substitution* behaviour of `predict` using its argument `method="plug-in"` (see `?predict-lda` for details). However, this type of prediction gives an over-optimistic assessment of the LDA classification (the same data are used to *train* the model and assess its performance). Ideally, we should use new samples (a *validation* or *test* data set) not used in building the LDA model. A statistical approach to resemble that is using *cross-validation* (see Resampling and Cross-Validation section below). The assessment of the performance of the model by cross-validation (CV) will be generally more realistic than those based on the same data used to fit the model (re-substitution). Different ways to perform cross-validation can be implemented. For example, with *leave-one-out cross-validation*, we leave out a sample, construct the LDA model and conduct prediction, then predict the class of the left-out sample. This is repeated for every sample in the data set. We can access to this cross-validation approach using the own `lda` function.

```
pods.lda.cv <- lda(Variety ~ Length + Width + Greenness + Curvature,
                   data = pods, prior = c(.25, .25, .25, .25), CV = T)
pods.lda.cv$class
```

```
 [1] C A A A A A A A A A B C B B B B B B B B B B C B B B C B B B D D D D D D D D
[39] D D
Levels: A B C D
```

The associated confusion table is the following.

```
addmargins(table(pods$Variety, pods.lda.cv$class))
```

```
       A  B  C  D Sum
  A    9  1  1  0  11
  B    0 10  1  0  11
  C    0  6  2  3  11
  D    0  0  0  7   7
  Sum  9 17  4 10  40
```

The cross-validated overall accuracy is reduced to 70%.

```
prop.table(table(pods$Variety, pods.lda.cv$class), 1)
```

```
         A      B      C      D
  A 0.8182 0.0909 0.0909 0.0000
  B 0.0000 0.9091 0.0909 0.0000
  C 0.0000 0.5455 0.1818 0.2727
  D 0.0000 0.0000 0.0000 1.0000
```

Note the poorer predictive performance suggested by these cross-validated calculations.

3.4.1 Class prediction for new (unknown) samples

We need to put the new samples into a `data.frame`. For illustration, suppose two new samples whose values are equal to those of the last two samples in the pea pods data file.

```
newdata <- rbind(c(73.99, 12.6, 1.2, 75.6), c(76.87, 12.6, 1.2, 72.4))
newdata <- data.frame(newdata)
# Create names for columns as in original data set
names(newdata) <- c("Length", "Width", "Curvature", "Greenness")
# Prediction
predict(pods.lda, newdata, dimen = 2)
```

```
$class
[1] D D
Levels: A B C D

$posterior
         A        B      C     D
1 3.01e-05 0.000020 0.0213 0.979
2 9.52e-05 0.000293 0.0958 0.904

$x
   LD1   LD2
1 2.76  0.25
2 2.30 -0.17
```

Both new samples have been allocated to variety D.

3.5 Case study: classification of bank notes

The data (Flury and Riedwyl, 1988) consist of six variables measured on 100 genuine and 100 counterfeit old Swiss bank notes. The first column contains the grouping variable (Genuine, Fake), the next ones correspond to the following 6 variables:

- Length: Length of the bank note.
- HeightL: Height of the bank note, measured on the left.
- HeightR: Height of the bank note, measured on the right.
- DistLB: Distance of inner frame to the lower border.
- DistUB: Distance of inner frame to the upper border.
- LengthD: Length of the diagonal.

Observations 1–100 are the genuine bank notes and the other 100 observations are the counterfeit bank notes. The aim is to study how these measurements may be used in determining whether a note is genuine or fake.

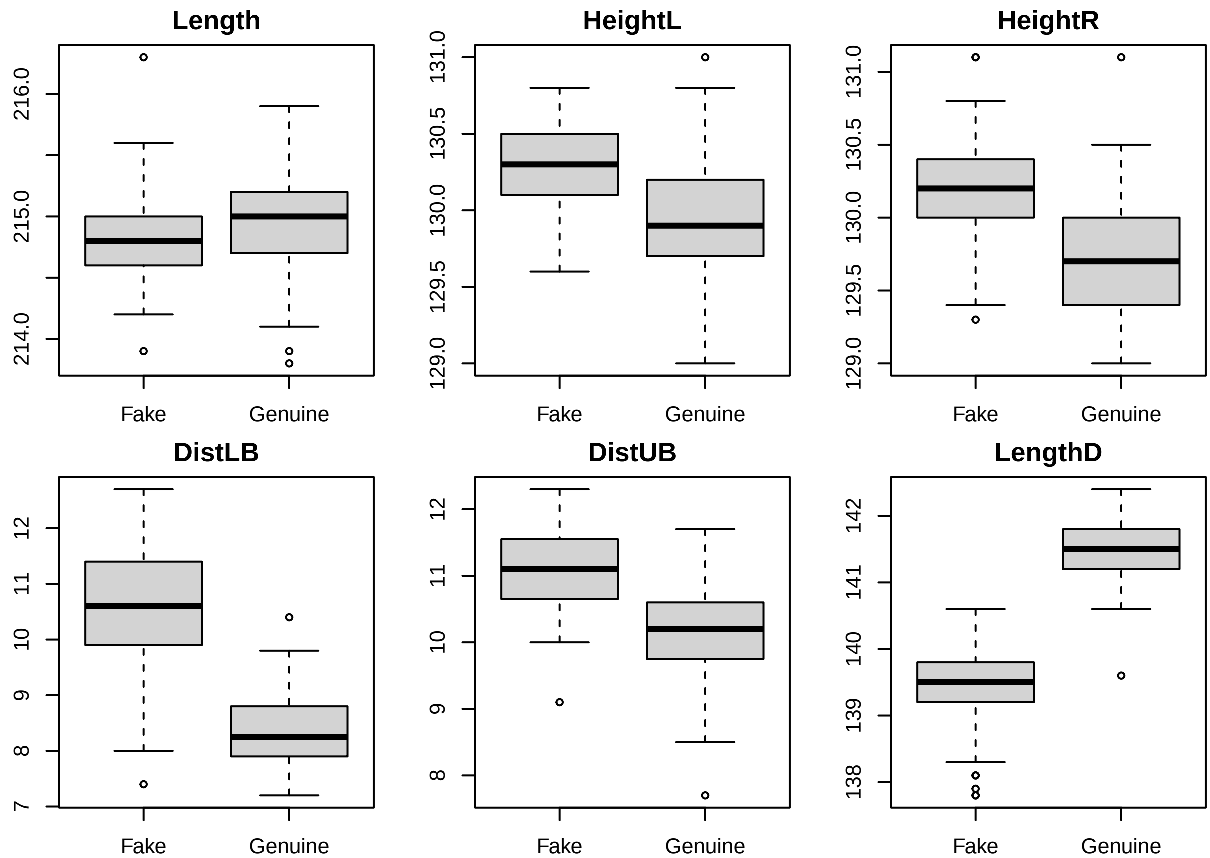
3.5.1 Preliminaries

1. Conduct exploratory inspection of the variables. What variables seem to be most helpful to distinguish groups?

Variables like LengthD and DistLB show the clearest differences between groups. Maybe Length will not be very helpful.

```
load("bank_notes.Rdata")
# Visual inspection by group
par(mfrow = c(2, 3)) # Boxplots arrangement
par(mar = c(2, 2, 2, 2))
```

```r
for (i in 2:ncol(bank)) {
  boxplot(bank[, i] ~ Group, data = bank, main = names(bank)[i])
}
```



2. Test for differences in mean between groups using multivariate analysis of variance (MANOVA). What does the result mean in relation to our aims?

```r
# Compare group means
bank.manova <- manova(cbind(Length, HeightL, HeightR, DistLB, DistUB, LengthD) ~ Group,
                      data = bank)
summary(bank.manova, test = "Hotelling-Lawley")
```

```
           Df Hotelling-Lawley approx F num Df den Df Pr(>F)
Group       1             12.2      392      6    193 <2e-16 ***
Residuals 198
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the MANOVA Hotelling-Lawley's test we conclude highly statistically significant differences in means ($p < 0.0001$). Hence, the selected variables seem to be useful to discriminate between groups.

```r
# Individual ANOVAs from the MANOVA result
summary.aov(bank.manova)
```

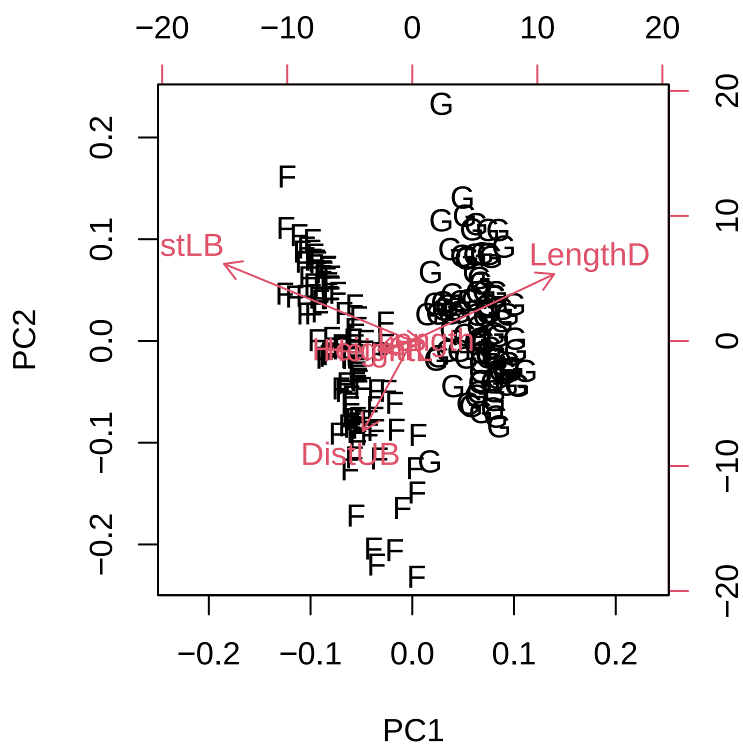
```
 Response Length :
             Df Sum Sq Mean Sq F value Pr(>F)
Group         1   1.07   1.066    7.77 0.0058 **
Residuals   198  27.15   0.137
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response HeightL :
             Df Sum Sq Mean Sq F value  Pr(>F)
Group         1   6.37    6.37    64.5 8.5e-14 ***
Residuals   198  19.57    0.10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response HeightR :
             Df Sum Sq Mean Sq F value Pr(>F)
Group         1   11.2   11.19     104 <2e-16 ***
Residuals   198   21.3    0.11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response DistLB :
             Df Sum Sq Mean Sq F value Pr(>F)
Group         1    248   247.5     292 <2e-16 ***
Residuals   198    168     0.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response DistUB :
             Df Sum Sq Mean Sq F value Pr(>F)
Group         1   46.6    46.6     113 <2e-16 ***
Residuals   198   81.7     0.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response LengthD :
             Df Sum Sq Mean Sq F value Pr(>F)
Group         1  213.6   213.6     836 <2e-16 ***
Residuals   198   50.6     0.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All comparisons per variable conclude statistically significant differences in mean between groups.

3. Using a biplot, can you identify any variable playing a leading role in distinguishing groups?

```r
# PCA analysis
biplot(prcomp(bank[, 2:7]), xlabs = substring(bank$Group, 1, 1))
```



Groups are neatly distinguished along the LengthD axis.

3.5.2 Linear discriminant analysis

Note that the discriminant scores (also called *canonical variates*) are usually centred to have mean zero. The coefficients of the linear discriminant functions are shown in the output and given by the component `scaling` of a `lda` object. They are associated to centred variables. So the discriminant scores provided by `lda` are obtained multiplying them by the centred original data and not the original data directly. You can use `str(bank.lda)` to see all the components of a `lda` object or check the help system out (`?lda`).

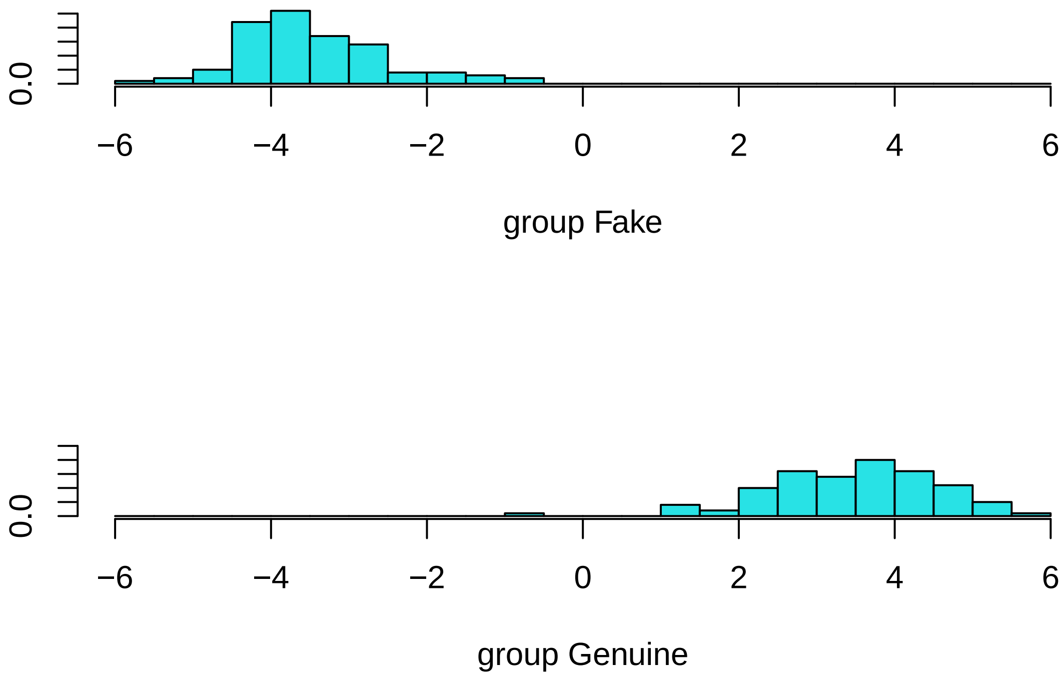
4. Perform LDA on the bank notes data and show the main results, including discriminant function coefficients, re-substitution and cross-validated confusion matrices.

```
library(MASS)
bank.lda <- lda(Group ~ ., data = bank) # Using formula interface: Group on all the others
bank.lda # Main output of the function.
```

```
Call:
lda(Group ~ ., data = bank)

Prior probabilities of groups:
   Fake Genuine
    0.5     0.5

Group means:
        Length HeightL HeightR DistLB DistUB LengthD
Fake       215     130     130   10.5   11.1     139
Genuine    215     130     130    8.3   10.2     142

Coefficients of linear discriminants:
             LD1
```

```
Length   0.00501
HeightL  0.83243
HeightR -0.84899
DistLB  -1.11734
DistUB  -1.17888
LengthD  1.55652
```

```r
plot(bank.lda) # Plots histogram of the discriminant scores by group
```



Predictions for each bank note (showing the first 6 only using the `head` function):

```r
head(predict(bank.lda)$x) # Discriminant scores
```

```
   LD1
1 2.15
2 4.59
3 4.58
4 4.75
5 4.21
6 2.65
```

```r
head(predict(bank.lda)$posterior) # Posterior probabilities for the groups
```

```
      Fake Genuine
1 3.25e-07       1
2 1.45e-14       1
3 1.54e-14       1
4 4.70e-15       1
5 1.94e-13       1
6 1.02e-08       1
```

```r
head(predict(bank.lda)$class) # allocated group
```

```
[1] Genuine Genuine Genuine Genuine Genuine Genuine
Levels: Fake Genuine
```

```r
# Confusion matrix based on re-substitution
# table(bank$Group, predict(bank.lda)$class)
# Maybe clearer editing the column names
pred.gr <- c("Predict Fake", "Predict Genuine")
table(bank$Group, factor(predict(bank.lda)$class, labels = pred.gr))
```

```
          Predict Fake Predict Genuine
  Fake             100               0
  Genuine            1              99
```

```r
# Confusion matrix based on leave-one-out cross-validation
# (no difference with raw confusion matrix in this case)
bank.lda.cv <- lda(Group ~ ., data = bank, CV = T)
table(bank$Group, factor(bank.lda.cv$class, labels = pred.gr))
```

```
          Predict Fake Predict Genuine
  Fake             100               0
  Genuine            1              99
```

5. Compute the correlation between discriminant scores and original variables as an approximation to their importance to distinguish the groups.

```r
cor(cbind(bank[, 2:7], predict(bank.lda)$x))
```

```
         Length HeightL HeightR DistLB  DistUB LengthD    LD1
Length   1.0000   0.231   0.152 -0.190 -0.0613   0.194  0.202
HeightL  0.2313   1.000   0.743  0.414  0.3623  -0.503 -0.516
HeightR  0.1518   0.743   1.000  0.487  0.4007  -0.516 -0.610
DistLB  -0.1898   0.414   0.487  1.000  0.1419  -0.623 -0.803
DistUB  -0.0613   0.362   0.401  0.142  1.0000  -0.594 -0.627
LengthD  0.1943  -0.503  -0.516 -0.623 -0.5940   1.000  0.935
LD1      0.2022  -0.516  -0.610 -0.803 -0.6267   0.935  1.000
```

Focusing on the last line for the discriminant scores (LD1), we can see that the results are consistent with what we obtained before.

How to classify a new observation using the model?

```r
# Classification of a new observation
new <- matrix(c(214.70, 131.12, 131.01, 10.63, 10.99, 140.13), ncol = 6) # New data
new <- as.data.frame(new) # Create data.frame ...
names(new) <- names(bank[, 2:7]) # ... with the same var names
predict(bank.lda, newdata = new) # Provides allocated group, posterior prob and score
```

```
$class
[1] Fake
Levels: Fake Genuine

$posterior
  Fake  Genuine
1    1 7.11e-08

$x
    LD1
1 -2.37
```

3.6 Resampling and cross-validation

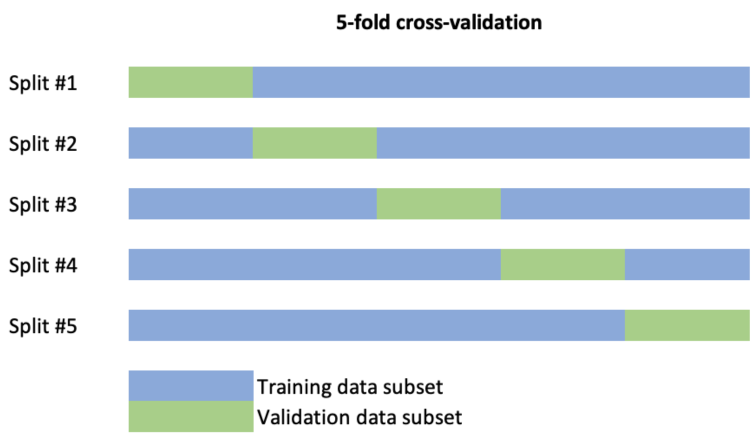
Resampling methods involve repeatedly drawing samples from a data set and recompute statistics or refit models of interest on each sample in order to obtain additional information. Cross-validation (CV) is a statistical resampling method commonly used to obtain estimates of the performance of predictive models from the available data. These estimates are generally more realistic than those based on the observed data set only (re-substitution). A data set is split into a *training data set* used to fit models and estimate parameters and a *validation data set* that does not participate in the fitting process and is used to assess the performance of the model. Sometimes an additional completely independent is also used for further performance assessment. This is then called a *test data set*, but often validation and test data set are used exchangeably.

There are quite a few ways to perform cross-validation:

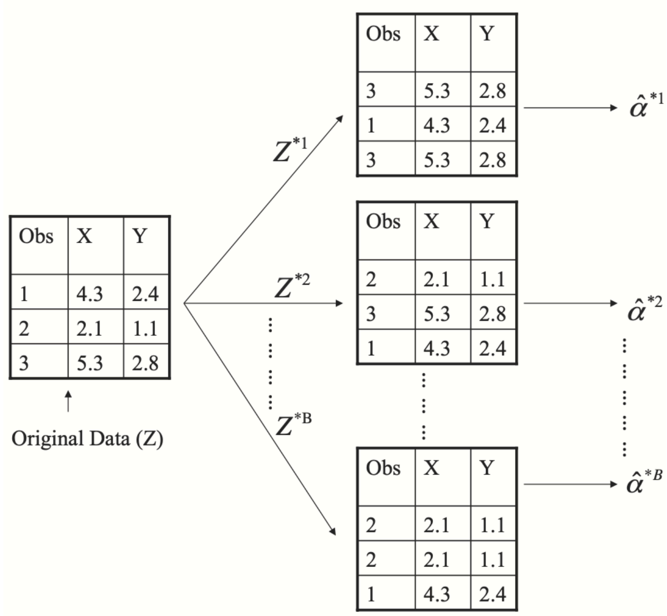
- K-fold cross-validation randomly divides the data into k blocks or splits of roughly equal size. Each of the blocks is left out in turn as validation data and the other $k-1$ blocks are used to train the model. The held-out block is predicted and these predictions are summarised into some type of performance measure. For example accuracy (overall proportion of cases rightly allocated) in a classification problem or the mean squared error (MSE),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

  in a regression problem. When k is equal to the sample size this procedure is known as *leave-one-out* CV (LOOCV). A schematic display of the splits of a data set to conduct 5-fold cross-validation is the following.



- Repeated k-fold CV does the same as above but more than once. For example, five repeats of 10-fold CV would give 50 total resamples. Note that this is not the same as 50-fold CV.
- Leave Group Out cross-validation (LGOCV), also known as Monte Carlo CV, randomly leaves out some set percentage of the data a large number of times.
- Bootstrap takes a random sample with replacement from the training set a large number of times. Since the sampling is with replacement, there is a strong likelihood that some training set samples will be represented more than once. As a consequence of this, some training set data points will not be contained in the bootstrap sample. The model is trained on the bootstrap sample and those data points not in that sample are predicted as hold-outs. Note that bootstrapping is actually generally used for statistical inference and data modelling for different purposes. The following figure illustrates the bootstrap procedure (extracted from James et al. (2021), An introduction to Statistical Learning, 2nd edition, Springer).

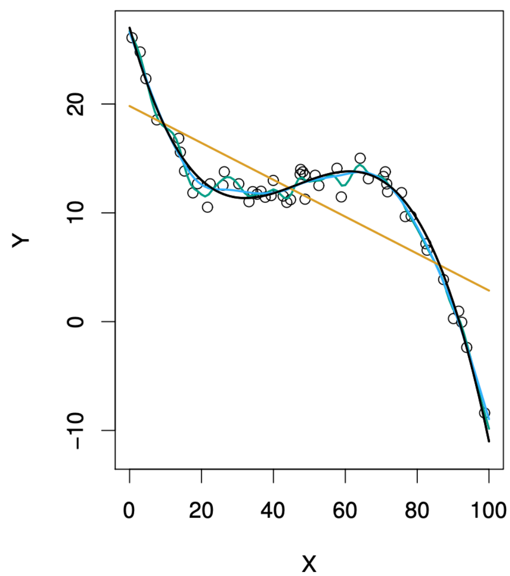


Which resampling or cross-validation method to use depends on the data set size and a some other factors. Each of the methods above can be characterised in terms of their *bias* and *precision*. This leads to what is called the *bias-variance* tradeoff, what is a recurring topic in statistical modelling and machine learning methods in general.

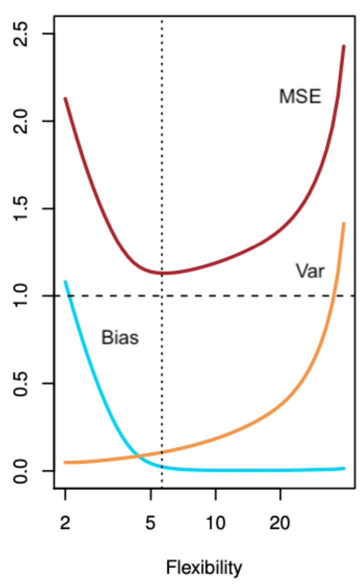
For example, when assessing the performance of a regression model we are interested in minimising the MSE, but MSE is decomposed as

$$MSE = Bias(\hat{y}, y)^2 + Var(\hat{y})$$

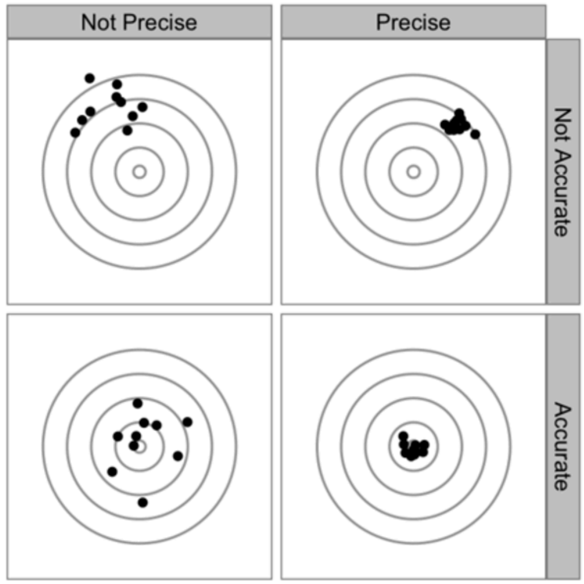
Hence, in order to minimise MSE, a method should simultaneously achieve low bias and low variance. Bias refers to the error, the deviation of the estimate from the actual value. Variance refers to how variable the estimate is, the amount by which the estimate would change using a different training data set. Ideally the estimate should not vary too much using different training data sets. In general, more flexible statistical methods have higher variance. For example, in the following figure we show a set of observed data (empty circles) and two fits to them: a linear fit (orange curve; using linear regression; strict modelling of the relationship between Y and X) and a non-linear fit (dark curve; using a smoothing spline; flexible modelling of the relationship between Y and X) (from James et al., 2021).



If the fit of the smoothing spline to the data is too tight, the bias will tend to be lower, but small changes in the training data can result in large changes in the estimates $\hat{y}$. Hence, the variance in the estimates or predictions will tend to be higher. On the other hand, linear regression is obviously too simplistic for these data. Thus, regardless of the training data, the estimation bias will be high. However, if the data were closer to linear, given enough data, linear regression would produce accurate predictions. The tradeoff comes from the fact that as more flexible is a method the variance tends to increase and the bias tends to decrease. Initially, the bias typically decreases faster than the variance increases, then leading to decrease in MSE. However, at some point increasing flexibility has little impact on the bias and the variance starts to increase significantly, leading to increase in MSE. In both the regression and classification modelling settings, choosing the correct level of flexibility is critical to the success of any method. The following figure illustrates this from simulated data (adapted from James et al., 2021).



The bias-variance tradeoff is analogously present in the context of cross-validation. Some types of CV have higher bias than others and the same is true for variance. The following figure summarises the discussion about bias (accuracy) versus variance (precision). The lower right is obviously the best place.



The bias and variance of a CV procedure is thought to be related to how much data is held out. For example, with LOOCV we repeatedly fit the statistical method using training sets that contain $n-1$ observations (nearly as many as in the entire data set). The MSE (computed from the validation or test set) is estimated as an average:

$$MSE_{CV} = \frac{1}{n}\sum_{i=1}^{n} MSE_i,$$

where MSE_i is the MSE obtained in each round of the LOOCV procedure. LOOCV tends to have lower bias than other methods. But it can be computationally expensive to implement because the model has to be fit n times, especially if n is large and if each individual model is slow to fit. As said above, using k-fold CV we split the set of observations into k groups (folds) with one of then used as validation set and the remaining $k-1$ as training set. This is computationally lighter than LOOCV as it is repeated $k < n$ times (producing k estimates of MSE which are averaged at the end of the process). However another potential advantage is that k-fold CV often has less variance than LOOCV, since there is less overlap between training sets. With LOOCV we are averaging the outputs of n fitted models, each of which is trained on an almost identical set of observations; therefore, these outputs are highly (positively) correlated with each other. The mean of many highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated. In exchange k-fold CV is expected to be more biased than LOOCV. If you hold-out 50% of your data using 2-fold CV, the thinking is that the final MSE estimate will be more biased than one that held out 10%. On the other hand, it is accepted that holding less data out (in the case of LOOCV just one sample) increases variance, since each hold-out sample has less data to get a stable estimate of performance. However, other factors are important. For example, if you have a ton of data, the bias and variance of 10- or even 5-fold CV may be acceptable. It has been shown empirically, that using $k = 5$ or $k = 10$ is a good compromise in general, yielding to errors that have neither excessively high bias nor very high variance (see also Kuhn and Johnson (2013), Applied Predictive Modeling, Springer for further discussion).

3.7 Exercises

1. PEA PODS data. To what extent are the scores on the canonical axes correlated? Multiply one of the variables by 10 and redo the discriminant analysis. What effect has this rescaling had on the loadings, scores and posterior probabilities? A slick way of doing this is to use the R function `I` ("Inhibit"), as follows.

```
pods.lda <- lda(Variety ~ Length + Width + I(10 * Greenness) + Curvature,
                data = pods, prior = c(.25, .25, .25, .25))
```

2. BRINE data. A set of geochemical measurements on brines from wells. Each sample is assigned to a stratigraphic unit, listed in the last column. How successfully can the geochemical measurements be used to predict stratigraphic unit?
3. PEA PODS data. Redo the discriminant analysis omitting the sample(s) you think might have been misclassified originally. Have the results changed in any way? To which variety does the discriminant analysis allocate the mis-classified sample(s)? To do this either create a copy of the data file and remove the sample(s) that seem to have been originally mis-classified or use the `subset` argument of `lda`.
4. SKULLS data. Carry out a discriminant analysis of the skulls data but before doing so check to see what kind of distributions the original variables have. What is the maximum number of canonical variates that can be derived? How many are worth using to discriminate among the epochs? What is the predictive ability of the selected canonical variates?

4 Correspondence analysis

4.1 Introduction

Correspondence analysis (CA) is a multivariate tool for analysing the associations between rows and columns of contingency tables, or tabular data in general. A contingency table is a two-way frequency table where the joint frequencies of two categorical variables are reported. The well-known test for independence, based on the χ^2 (chi-square) statistic, is useful to investigate statistically significant association. However there are no simple ways for detecting which parts of the table are responsible for this association. CA is one tool that allows to see the pattern of association in the data and to generate hypotheses that can be tested in a subsequent stage of research.

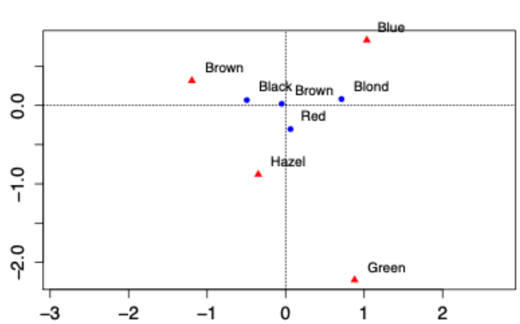
CA reduces the dimension of a contingency table in order to obtain a graphical representation of rows and columns in a low-dimensional space. The idea is to extract new dimensions (axis) in decreasing order of importance so that the main information of the table can be summarised in spaces with low dimensions. For example, if only two axis are used, the results can be shown on a two-dimensional plane. This is why CA is considered as a PCA for categorical variables. With PCA, the total variance is partitioned into independent contributions from the principal components. CA, instead, decomposes a measure of association, particularly the χ^2 statistic. Another difference is that CA works on count/frequency data, whereas PCA is thought for continuous data. Actually, CA can work on any non-negative discrete data as long as they are measured on the same scale, so that it makes sense to compute row sums and column sums.

Hence, the variables of interest can also be discrete quantitative variables, such as the number of family members or the number of accidents an insurance company had to cover during one year, etc. Here, each possible value that the variable can have defines a row or a column category. Of course this will be practically feasible as long as the number of different values observed is low-moderate. Continuous variables may be taken into account by discretisation, i.e. defining categories in terms of intervals or classes of values which the variable can take on. Thus contingency tables can be used in many situations, implying that correspondence analysis is a very useful tool in many applications.

Example: Hair and eye colour.

		Eye				
		Brown	Blue	Hazel	Green	TOTAL
Hair	Black	32	11	10	3	56
	Brown	53	50	25	15	143
	Red	10	10	7	7	34
	Blond	3	30	5	8	46
	TOTAL	98	101	47	33	279

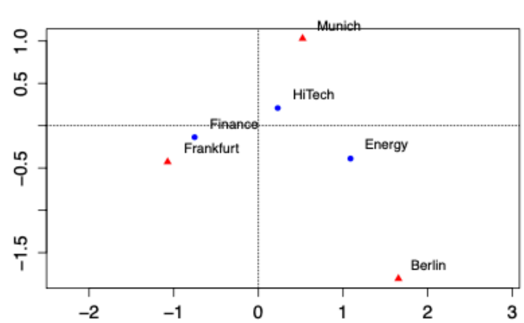
Correspondence analysis map:



Example: Companies and locations.

		City		
		Frankfurt	Berlin	Munich
Sector	Finance	4	0	1
	Energy	0	1	1
	HiTech	2	1	4

Correspondence analysis map:



4.2 Basic elements

Let $\mathbf{N}$ denote the $I \times J$ contingency table with entries n_{ij} and positive row and column sums. The matrix $\mathbf{N}$ can be converted to the matrix *correspondence matrix* $\mathbf{P} = [p_{ij}]$ by dividing $\mathbf{N}$ by the grand total $n = \sum_{i=1}^{I} \sum_{j=1}^{J} n_{ij}$. The vectors of marginal relative frequencies by rows and columns are called *row* and *column masses* with components

$$r_i = \sum_{j=1}^{J} p_{ij} \quad \text{and} \quad c_j = \sum_{i=1}^{I} p_{ij}$$

All these quantities $\mathbf{P}$, $\mathbf{r} = (r_1, \dots, r_I)$ and $\mathbf{c} = (c_1, \dots, c_J)$ are proportions adding up to 1 in each case. Besides, the *row* and *column profiles* are defined as the row (column) frequencies divided by their row (column) total. There is a complete symmetry in CA analysis in that the results are the same regardless of the main interest being on the rows or on the columns of the table.

The well-known Pearson's χ^2 test for independence measures the discrepancy between all the observed, n_{ij}, and expected, $e_{ij} = nr_ic_j$, frequencies for each cell (i, j) of the table using the χ^2 statistic:

$$\chi^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(n_{ij} - e_{ij})^2}{e_{ij}} = n \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(p_{ij} - r_ic_j)^2}{r_ic_j}$$

which has distribution $\chi^2_{(I-1)(J-1)}$ under independence. The larger this value, the more discrepant the observed and expected frequencies are, i.e., the less convincing the assumption of independence of rows and columns is.

The quantity χ^2/n, called *total inertia* or simply inertia, plays a central role in CA as a measure of variation in a data table. This is equivalent to the total variability in PCA.

We seek to represent the row categories by scores which approximate the distances/similarities between the row profiles of the table and scores for the columns which approximate the distances/similarities between the columns of the table, both after a little scaling and aiming to maximise the explained data variability (total inertia). These scores are given by the set of first dimensions for the rows and columns of the CA solution, as for PCA.

We can proceed to generate second dimensions for the rows and columns in a manner similar to the way in which a second principal component is derived in PCA. And so on for further dimensions, each dimension describing less of the association between the rows and columns than the previous dimensions.

Typically, we plot the row and column scores for the first two dimensions in the same plot to produce a CA biplot or map.

4.3 CA of the moths data set

The moths data set shows the counts of 12 different moth species recorded in light traps at 14 sites throughout the UK. For each site we also have its Northing, Easting and Type (Woodland, Farmland, Parkland).

```
moths <- read.table("moths.txt", h = T, stringsAsFactors = TRUE)
moths
```

```
  Site   E   N Type  M1  M2 M3  M4  M5 M6  M7  M8  M9 M10 M11 M12
1    A 513 213    W 154  51  0  37  20 55   1   5   1 724   0   0
2    B 465 164    W  34  57  5  66   8 21 141  33  35  84   0   0
3    C 480 143    W   5  26  5  30   2 20  12  66 214  73   0   2
4    D 418 107    W   0  37 55   5   3 63   7  23   6   0   0   0
5    E 306 138    P  31 109  6  12  23 56   9 403  58   0   2  13
```

```
6      F 282  45    F   1  40  1  79  66 35  77  35   6   0   0   0
7      G 263 284    F   5  14  2  27 144 54  24   7   7   2   0  14
8      H 520 280    P   3  17  0 316  79 19  21   4  48   7   0   0
9      I 575 266    F   0   7  0 126  37 10 101  83   3   0   0   0
10     J 363 594    W  64 383  1   0  47  0   0   0   0  79  65   0
11     K 260 756    P  26 373  2   0 228  0   0   0   0 129 393   0
12     L 237 809    W   8 306 94   2 108  0   0   7   0   0  55   0
13     M 316 864    W   6 141  0   6  38  0   9   0   0   0  10   0
14     N 263 874    W   9 535  0   0 156  0   0   0   0 429 141   6
```

```r
# Extract the counts and use site as row label
counts <- data.frame(row.names = moths$Site, moths[5:16])
```

The following computes the CA solution using the R package `ca` (the function `corresp` in the MASS package is an alternative, although it has less options). The output provides, amongst others, information about the amount of explained variability by the successive CA dimensions (so-called *principal inertias*, technically corresponding to eigenvalues) and the scores for rows and columns, i.e. their coordinates in the low-dimensional representation. The first two dimensions explain 47.9% of the total variability/inertia.

```r
library(ca)
moths.ca <- ca(counts)
options(digits = 3)
moths.ca
```

```
 Principal inertias (eigenvalues):
           1        2        3        4      5        6        7
Value      0.604647 0.377591 0.349915 0.2102 0.186407 0.111768 0.100981
Percentage 29.5%    18.42%   17.07%   10.26% 9.1%     5.45%    4.93%
           8        9        10       11
Value      0.066095 0.023946 0.016025 0.001933
Percentage 3.22%    1.17%    0.78%    0.09%


 Rows:
             A       B       C     D      E      F      G      H      I       J
Mass     0.126  0.0584  0.0549 0.024 0.0871 0.0410 0.0362  0.062 0.0443  0.0771
ChiDist  1.484  1.2371  2.0952 2.356 1.8430 1.2140 1.6048  1.976 1.6400  0.9585
Inertia  0.279  0.0894  0.2411 0.133 0.2960 0.0605 0.0933  0.242 0.1191  0.0709
Dim. 1  -0.601  0.8334  1.3013 0.564 1.3059 1.0849 0.4967  1.483 1.6599 -0.8795
Dim. 2  -2.192 -0.5295 -0.7443 1.117 0.5151 0.3490 0.6063 -0.112 0.1929  0.3665
             K      L       M       N
Mass     0.139  0.070  0.0253  0.1540
ChiDist  1.134  1.338  1.0771  0.7718
Inertia  0.179  0.125  0.0294  0.0917
Dim. 1  -0.967 -0.665 -0.6080 -0.8787
Dim. 2   0.793  1.444  0.9772 -0.1612


 Columns:
             M1     M2      M3      M4      M5      M6      M7     M8      M9
Mass     0.0418  0.253  0.0206  0.0852  0.1158  0.0402  0.0485 0.0804  0.0456
ChiDist  1.1587  0.802  2.7702  1.8725  0.8636  1.4941  1.9031 1.9738  2.3538
Inertia  0.0561  0.163  0.1584  0.2988  0.0863  0.0897  0.1757 0.3132  0.2528
Dim. 1  -0.3998 -0.709 -0.1035  1.5703 -0.1920  0.8832  1.3980 1.6030  1.6069
Dim. 2  -1.3178  0.689  1.8790 -0.1958  0.7470 -0.0125 -0.0242 0.4834 -0.6108
```

```
           M10     M11     M12
Mass     0.184  0.0804 0.00422
ChiDist  1.212  1.4239 2.29173
Inertia  0.271  0.1630 0.02219
Dim. 1  -0.699 -1.1606 0.78124
Dim. 2  -1.730  0.9842 0.59183
```

```r
moths.ca$rowcoord[, 1:2] # Row coordinates in the first two CA dims
```

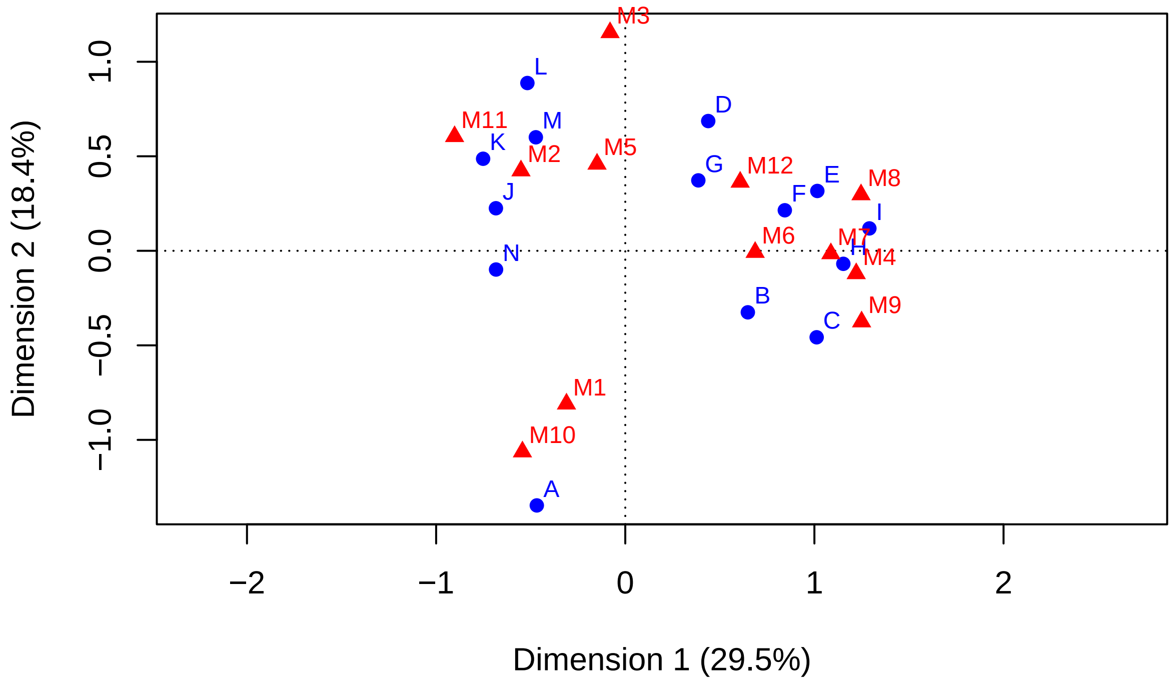
```
    Dim1   Dim2
A -0.601 -2.192
B  0.833 -0.529
C  1.301 -0.744
D  0.564  1.117
E  1.306  0.515
F  1.085  0.349
G  0.497  0.606
H  1.483 -0.112
I  1.660  0.193
J -0.879  0.367
K -0.967  0.793
L -0.665  1.444
M -0.608  0.977
N -0.879 -0.161
```

```r
moths.ca$colcoord[, 1:2] # Column coordinates in the first two CA dims
```

```
      Dim1    Dim2
M1  -0.400 -1.3178
M2  -0.709  0.6891
M3  -0.103  1.8790
M4   1.570 -0.1958
M5  -0.192  0.7470
M6   0.883 -0.0125
M7   1.398 -0.0242
M8   1.603  0.4834
M9   1.607 -0.6108
M10 -0.699 -1.7297
M11 -1.161  0.9842
M12  0.781  0.5918
```

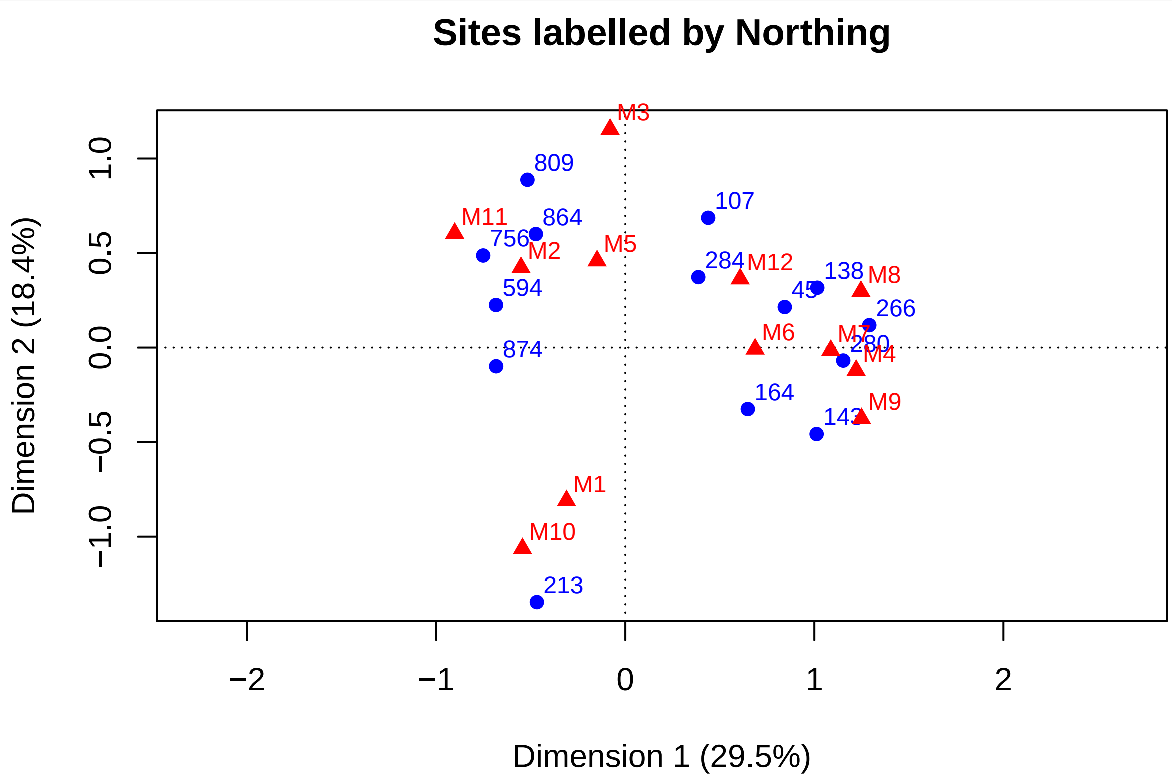
The CA map (which is a biplot as it represents two sets of points, rows and column coordinates) is not particularly revealing.

```r
plot(moths.ca)
```



But labelling the sites by their Northing reveals that the first dimension is mainly a north/south axis and that species 2 and 11 do well at these northerly sites.

```
# Re-label sites using Northing
rownames(counts) <- moths$N
moths.ca2 <- ca(counts)
plot(moths.ca2, main = "Sites labelled by Northing")
```



In general, when both the Site and Species scores sit close to each other and far from the origin this implies they have a strong association. The count for that species on that site is greater than you would expect if there was no association. For example, species 2 and 11 in the CA plot sit close to sites J, K, L and M. This is because those species have particularly high (negative) scores at those sites which themselves have high (negative) scores, both in the first dimension. The same is true, but with positive scores, for Species 4 and 7 at sites H and I.

A CA map is interpreted as follows:

1. Proximity of two row (or two column) scores indicates similar, proportional row (column) profiles across columns (across rows).
2. Proximity between rows and column scores reflects on positive association between them, and the opposite implies negative association.
3. If all row and column scores are very close to the origin, this implies poor association between them (typically obtaining a no statistically significant result from a χ^2-test of independence).
4. The origin $(0, 0)$ represents the average of the row and column profiles. Hence, a particular point projected close to the origin indicates closeness to the average profile.
5. All the interpretations must be carried out in view of the quality of the representation, which is given by the cumulated variability/inertia explained by the two first principal axes/CA dimensions.

We have used above what is formally called a *symmetric* CA map, but there are some (*asymmetric*) variants (essentially modifications of the axis scaling) which may more precisely represent relationships between rows and columns. See e.g. Greenacre (2007) "Correspondence Analysis in Practice" for more details.

4.4 Case study 1: funding and disciplines

The data set consists of a cross-classification table in which counts from $N = 796$ scientific researchers were distributed according to their scientific discipline (geology, biochemistry, chemistry, zoology, physics, engineering, microbiology, botany, statistics and mathematics) and funding category (A, B, C, D, E). A to D are the categories for researchers who are receiving research grants, from A (most funded) to D (least funded), while E is a category assigned to researchers whose grant applications were not successful (source: Greenacre, 2007).

4.4.1 Contingency table, profiles, masses

```r
load("Funding.Rdata")
ct <- as.table(as.matrix(funding)) # create table object
ct
```

```
      A  B  C  D  E
Geol  3 19 39 14 10
Bioc  1  2 13  1 12
Chem  6 25 49 21 29
Zool  3 15 41 35 26
Phys 10 22 47  9 26
Engi  3 11 25 15 34
Micr  1  6 14  5 11
Bota  0 12 34 17 23
Stat  2  5 11  4  7
Math  2 11 37  8 20
```

```r
# Correspondence matrix and row and column masses
round(addmargins(ct / sum(ct)), 3)
```

```
         A     B     C     D     E   Sum
Geol 0.004 0.024 0.049 0.018 0.013 0.107
Bioc 0.001 0.003 0.016 0.001 0.015 0.036
Chem 0.008 0.031 0.062 0.026 0.036 0.163
Zool 0.004 0.019 0.052 0.044 0.033 0.151
Phys 0.013 0.028 0.059 0.011 0.033 0.143
Engi 0.004 0.014 0.031 0.019 0.043 0.111
Micr 0.001 0.008 0.018 0.006 0.014 0.046
Bota 0.000 0.015 0.043 0.021 0.029 0.108
Stat 0.003 0.006 0.014 0.005 0.009 0.036
Math 0.003 0.014 0.046 0.010 0.025 0.098
Sum  0.039 0.161 0.389 0.162 0.249 1.000
```

```r
# Row profiles
round(prop.table(ct, 1), 3)
```

```
         A     B     C     D     E
Geol 0.035 0.224 0.459 0.165 0.118
Bioc 0.034 0.069 0.448 0.034 0.414
Chem 0.046 0.192 0.377 0.162 0.223
Zool 0.025 0.125 0.342 0.292 0.217
Phys 0.088 0.193 0.412 0.079 0.228
Engi 0.034 0.125 0.284 0.170 0.386
Micr 0.027 0.162 0.378 0.135 0.297
Bota 0.000 0.140 0.395 0.198 0.267
Stat 0.069 0.172 0.379 0.138 0.241
Math 0.026 0.141 0.474 0.103 0.256
```

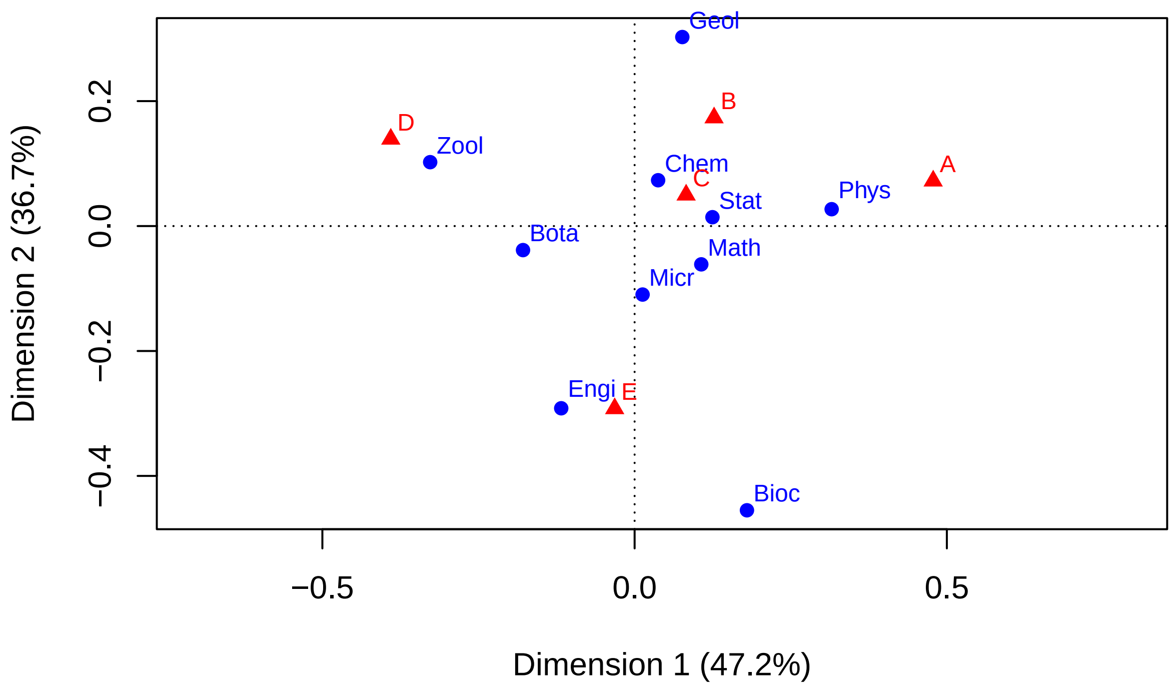
```r
# Column profiles
round(prop.table(ct, 2), 3)
```

```
         A     B     C     D     E
Geol 0.097 0.148 0.126 0.109 0.051
Bioc 0.032 0.016 0.042 0.008 0.061
Chem 0.194 0.195 0.158 0.163 0.146
Zool 0.097 0.117 0.132 0.271 0.131
Phys 0.323 0.172 0.152 0.070 0.131
Engi 0.097 0.086 0.081 0.116 0.172
Micr 0.032 0.047 0.045 0.039 0.056
Bota 0.000 0.094 0.110 0.132 0.116
Stat 0.065 0.039 0.035 0.031 0.035
Math 0.065 0.086 0.119 0.062 0.101
```

4.4.2 Correspondence analysis

We focus on the row profiles; that is, the main interest is on research areas and their relationship with the funding categories. Note that the ca function also admits the data.frame as input directly. That is, no need to create a table object first

```r
library(ca)
ct.ca <- ca(ct)
plot(ct.ca)
```



Looking at the relative position of the research categories, we can see that the first axis (horizontal) lines up the four categories of funding in their intrinsic ordering, from D (less funding) to A (most funding). The second axis, is a contrast between E (no funding) against the others. Hence, the more a discipline is low down in the map the less funding is granted. The more to the right, the more the funding received. The best place to be would be the top right quadrant. Physics is the most to the right, it shows the highest percentage of type A researchers (8.8%, see row profiles table). But it is at the middle vertically since it has a percentage of non-funded researchers close to average (22.8% compared to 24.9%, see row profiles and masses in correspondence matrix above). Biochemistry is the discipline with the highest number of rejections in relative terms. Note that statements about the funding profiles of the disciplines must all be in relative terms. For absolute differences one should refer to the original data.

Geology, statistics, mathematics and biochemistry are all at a similar position on the first axis, but widely different on the second. This means that the researchers in these fields whose grants have been accepted have similar positions with respect to the funded categories A to D categories, but geology has much fewer rejections than biochemistry.

4.4.3 Decomposition of the inertia

```r
summary(ct.ca)
```

```
Principal inertias (eigenvalues):

 dim    value      %   cum%   scree plot
 1      0.039117  47.2  47.2  ************
 2      0.030381  36.7  83.9  *********
 3      0.010869  13.1  97.0  ***
 4      0.002512   3.0 100.0  *
        -------- -----
 Total: 0.082879 100.0
```

```
Rows:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr  
1  | Geol |  107  916  137 |   76  55  16 |  303 861 322 |
2  | Bioc |   36  881  119 |  180 119  30 | -455 762 248 |
3  | Chem |  163  644   21 |   38 134   6 |   73 510  29 |
4  | Zool |  151  929  230 | -327 846 413 |  102  83  52 |
5  | Phys |  143  886  196 |  316 880 365 |   27   6   3 |
6  | Engi |  111  870  152 | -117 121  39 | -292 749 310 |
7  | Micr |   46  680   10 |   13   9   0 | -110 671  18 |
8  | Bota |  108  654   67 | -179 625  88 |  -39  29   5 |
9  | Stat |   36  561   12 |  125 554  14 |   14   7   0 |
10 | Math |   98  319   56 |  107 240  29 |  -61  79  12 |

Columns:
    name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr  
1 |    A |   39  587  187 |  478 574 228 |   72  13   7 |
2 |    B |  161  816  110 |  127 286  67 |  173 531 159 |
3 |    C |  389  465   94 |   83 341  68 |   50 124  32 |
4 |    D |  162  968  347 | -390 859 632 |  139 109 103 |
5 |    E |  249  990  262 |  -32  12   6 | -292 978 699 |
```

The first part of the output of **summary** shows that the 10×5 table produces 4 principal axes (dimensions). The two first (used in the CA map) explain 83.9% of the total inertia. The value of the χ^2 statistic is $\textit{total inertia} \times n = 65.97$.

The subsequent output provides more comprehensive details about how the inertia/variability is decomposed by rows and columns, the contributions of these to each principal axis, and how well they are represented in the CA map (we refer to Greenacre (2007)'s book for more details about these measures).

4.5 Case study 2: marine species abundance

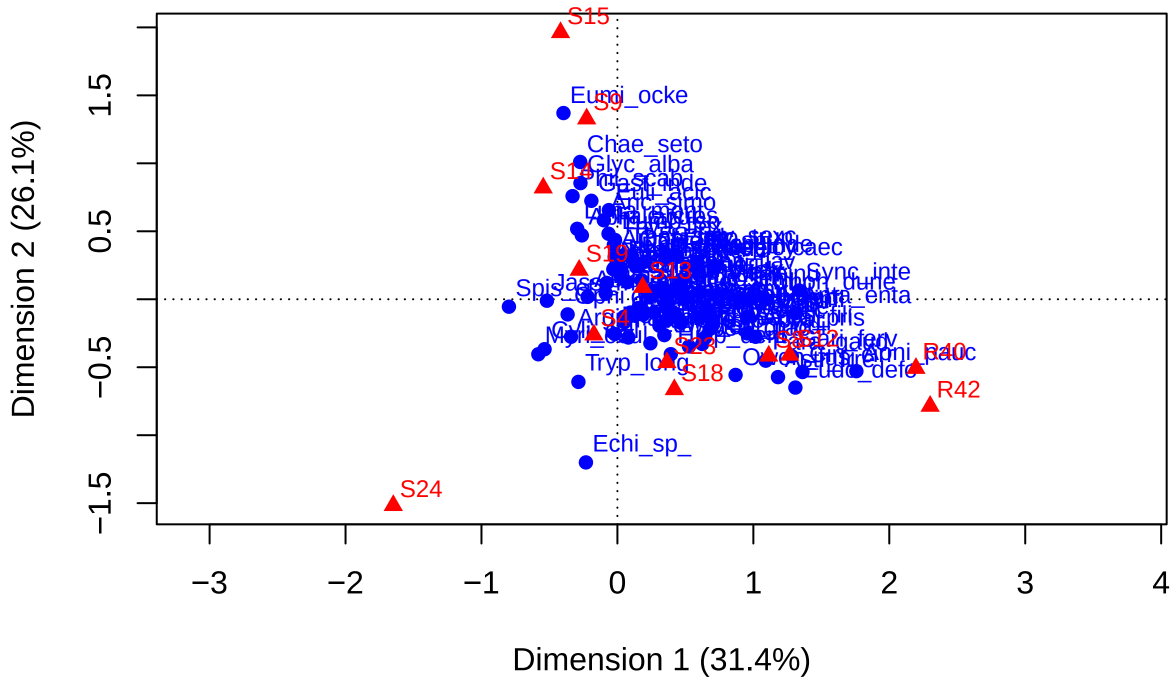
The data set consists of abundance (frequencies) of 92 marine species in 13 stations (samples) in the North sea

```r
species <- read.table("SpecAbund.txt", h = T, stringsAsFactors = TRUE)
# Show first lines of the table
head(species)
```

```
    Species  S4 S8  S9 S12 S13 S14 S15 S18 S19 S23 S24 R40 R42
1 Myri_ocul 193 79 150  72 141 302 114 136 267 271 992   5  12
2 Chae_seto  34  4 247  19  52 250 331  12 125  37  12   8   3
3 Amph_falc  49 58  66  47  78  92 113  38  96  76  37   0   5
4 Myse_bide  30 11  36  65  35  37  21   3  20 156  12  58  43
5 Goni_macu  35 39  41  37  32  45  41  41  31  29  64  32  23
6 Amph_fili  19 39  11  38  18  20  11  22  30  40   3  55  65
```

Stations starting with S are close to oil-drill platform (pollution), stations R40 and R42 are regarded as clean reference samples.

```r
# Extract the counts and use site as row label
sp.counts <- data.frame(row.names = species$Species, species[2:14])
sp.ca <- ca(sp.counts)
plot(sp.ca, map = "rowprincipal") # Use different scaling to facilitate interpretation
```



- Stations form a curve from most polluted stations to least polluted stations. Reference clean stations (R40, R42) are far from the drilling area.
- Station 24 separates out from the others: higher relative abundance of *Echi.sp* and *Myri.ocul.* here. Species *Eumi.oke.* is linked to the most polluted stations.
- The two first CA dimensions explain 57.5% of the total variation.

4.6 Exercises

1. MARKETING data. Do the marketing data show any association between breakfast food type and the housewives' assessments of them?

5 Relationships among samples

5.1 Introduction

Principal component analysis and discriminant analysis have analysed the data through the variables (columns) of the data set, extracting information from the covariance or correlation matrix of the variables. However, for other research purposes we switch focus away from relationships among the variables to relationships among the samples, analysing the data set $\mathbf{X}$ through the samples (rows). For this, instead of covariance matrices for variables, we start with similarity (or distance) matrices for samples.

5.2 Similarity and distance matrices

The *similarity* between two samples is often defined as a number in the range $[0, 1]$ where 1 implies identical and 0 implies nothing in common. The *distance* between two samples is the complement of similarity though distances are not always restricted to lie in the range $[0, 1]$. When they are restricted to this range there is often a simple relationship between similarity and distance such as $d = 1 - s$ or $d = 1 - s^2$. Distances are also referred to as *dissimilarities*.

If s_{ij} denotes the similarity between the i*th* and j*th* sample then the collection of all pairwise similarities between samples can be written as a matrix, referred to as the similarity matrix, S. The full n by n matrix, where n is the number of samples, will have 1s on all the diagonal positions because the similarity of a sample with itself must be 1. Also $s_{ij} = s_{ji}$.

In the same way, we can define a distance matrix D with individual elements d_{ij} and 0s on the diagonal.

From this point onwards we will be exploring methods of analysing multivariate data for which distance or similarity matrices are sometimes the natural starting point. Note that distance or similarity matrices are symmetrical and typically only the lower triangle is shown and stored. There is a wide range of similarity and distance measures, some generic (e.g. simple matching) and some more specialised (e.g. Nei-Li for molecular markers or the Bradley-Curtis which is popular in ecology). It is then essential to know how these measures are calculated, actually the details of a measure under the same name often change from one software package to another. When the data set includes several types of variables, say quantitative, binary and categorical, a strategy consists of defining an appropriate measure for each type of variables separately and the compute an overall similarity/distance measure (e.g. using Gower's formula).

Here we explore different methods of calculating such matrices for data of different types. We use the moths data set as reference to illustrate. It counts the number of moths in each of 12 species (variables) at each of 14 sites (samples).

5.3 Binary data

We begin by converting to the moths counts to presence/absence indicators. This involves a little programming.

```
moths <- read.table("moths.txt", h = T, stringsAsFactors = TRUE)
#copy in preparation for conversion to presence/absence (pa) format
mothspa <- moths
#convert the counts to 1=present, 0=absent
mothspa[, 5:16] <- ifelse(mothspa[, 5:16] > 0, 1, 0)
mothspa
```

```
   Site   E   N Type M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12
1     A 513 213    W  1  1  0  1  1  1  1  1  1   1   0   0
2     B 465 164    W  1  1  1  1  1  1  1  1  1   1   0   0
3     C 480 143    W  1  1  1  1  1  1  1  1  1   1   0   1
```

```
4      D 418 107    W  0  1  1  1  1  1  1  1  1   0   0   0
5      E 306 138    P  1  1  1  1  1  1  1  1  1   0   1   1
6      F 282  45    F  1  1  1  1  1  1  1  1  1   0   0   0
7      G 263 284    F  1  1  1  1  1  1  1  1  1   1   0   1
8      H 520 280    P  1  1  0  1  1  1  1  1  1   1   0   0
9      I 575 266    F  0  1  0  1  1  1  1  1  1   0   0   0
10     J 363 594    W  1  1  1  0  1  0  0  0  0   1   1   0
11     K 260 756    P  1  1  1  0  1  0  0  0  0   1   1   0
12     L 237 809    W  1  1  1  1  1  0  0  1  0   0   1   0
13     M 316 864    W  1  1  0  1  1  0  1  0  0   0   1   0
14     N 263 874    W  1  1  0  0  1  0  0  0  0   1   1   1
```

Considering sites K and L only. We create a `data.frame` from the K and L site data and manipulate it to compute a table showing the co-occurrence of moths across species in the two sites.

```r
KL <- data.frame(t(mothspa[mothspa$Site %in% c("K", "L"), 5:16]))
names(KL) <- c("K", "L")
KL[, 1] <- factor(KL[, 1], levels = c(1, 0), labels = c("Present", "Absent"))
KL[, 2] <- factor(KL[, 2], levels = c(1, 0), labels = c("Present", "Absent"))
ftable(KL[, c("L", "K")])
```

```
        K Present Absent
L                       
Present           5      2
Absent            1      4
```

The *simple matching* coefficient of similarity is $S_{KL} = (5+4)/(5+2+1+4) = 0.75$. The *Jaccard* coefficient of similarity (ignores "double zeros") is $S_{KL} = 5/(5+2+1) = 0.625$.

The package `vegan` has a wide range of options for distance coefficients, though its nomenclature is somewhat challenging. The function `vegdist` in `vegan` computes distances d, hence similarities can be obtained as $s = 1 - d$. The following code snippet computes the simple matching similarity between all sites using `vegan`.

```r
library(vegan)
# simple matching similarity matrix
sm <- 1 - vegdist(mothspa[5:16], method = "gower")
options(digits = 2)
sm
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.92                                                            
3  0.83 0.92                                                       
4  0.75 0.83 0.75                                                  
5  0.67 0.75 0.83 0.75                                             
6  0.83 0.92 0.83 0.92 0.83                                        
7  0.83 0.92 1.00 0.75 0.83 0.83                                   
8  1.00 0.92 0.83 0.75 0.67 0.83 0.83                              
9  0.83 0.75 0.67 0.92 0.67 0.83 0.67 0.83                         
10 0.42 0.50 0.42 0.33 0.42 0.42 0.42 0.42 0.25                    
11 0.42 0.50 0.42 0.33 0.42 0.42 0.42 0.42 0.25 1.00               
12 0.50 0.58 0.50 0.58 0.67 0.67 0.50 0.50 0.50 0.75 0.75          
13 0.58 0.50 0.42 0.50 0.58 0.58 0.42 0.58 0.58 0.67 0.67 0.75     
14 0.42 0.33 0.42 0.17 0.42 0.25 0.42 0.42 0.25 0.83 0.83 0.58 0.67
```

The following computes the matrix of Jaccard distances (the `round` function takes care of the number of decimal places).

```
# Jaccard distance matrix
jacc <- vegdist(mothspa[5:16], method = "jaccard")
round(jacc, 2)
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.10
3  0.18 0.09
4  0.30 0.20 0.27
5  0.33 0.25 0.17 0.27
6  0.20 0.10 0.18 0.11 0.18
7  0.18 0.09 0.00 0.27 0.17 0.18
8  0.00 0.10 0.18 0.30 0.33 0.20 0.18
9  0.22 0.30 0.36 0.12 0.36 0.22 0.36 0.22
10 0.64 0.55 0.58 0.73 0.58 0.64 0.58 0.64 0.82
11 0.64 0.55 0.58 0.73 0.58 0.64 0.58 0.64 0.82 0.00
12 0.55 0.45 0.50 0.50 0.36 0.40 0.50 0.55 0.60 0.38 0.38
13 0.50 0.55 0.58 0.60 0.45 0.50 0.58 0.50 0.56 0.50 0.50 0.38
14 0.64 0.67 0.58 0.83 0.58 0.75 0.58 0.64 0.82 0.29 0.29 0.56 0.50
```

Note that the function `dist` in the R base distribution also includes several distance measures (check out `?dist`). For example, the Jaccard distance matrix we just obtained can be equally computed using `dist` as

```
# Jaccard distance matrix using dist in R base
jacc <- dist(mothspa[5:16], method = "binary")
round(jacc, 2)
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.10
3  0.18 0.09
4  0.30 0.20 0.27
5  0.33 0.25 0.17 0.27
6  0.20 0.10 0.18 0.11 0.18
7  0.18 0.09 0.00 0.27 0.17 0.18
8  0.00 0.10 0.18 0.30 0.33 0.20 0.18
9  0.22 0.30 0.36 0.12 0.36 0.22 0.36 0.22
10 0.64 0.55 0.58 0.73 0.58 0.64 0.58 0.64 0.82
11 0.64 0.55 0.58 0.73 0.58 0.64 0.58 0.64 0.82 0.00
12 0.55 0.45 0.50 0.50 0.36 0.40 0.50 0.55 0.60 0.38 0.38
13 0.50 0.55 0.58 0.60 0.45 0.50 0.58 0.50 0.56 0.50 0.50 0.38
14 0.64 0.67 0.58 0.83 0.58 0.75 0.58 0.64 0.82 0.29 0.29 0.56 0.50
```

There are many other similarity coefficients for binary data and it is simply a matter of selecting the most appropriate for your circumstances. A literature review in your area is useful to get insight into what other people is using and what might be a good choice for the characteristics of your data.

5.4 Numeric data

The raw counts for the moths data are strongly skewed. A common strategy to reduce this effect is to take logs (after adding 1 to the counts to avoid problems with zeros), we use log base 10 as an example here.

```
mothslog <- moths # Create a separate data set
mothslog[, 5:16] <- log10(mothslog[, 5:16] + 1) # Transform the counts
options(digits = 2)
mothslog
```

```
   Site    E    N Type    M1   M2    M3    M4    M5   M6   M7    M8    M9   M10   M11   M12
```

```
1      A 513 213    W 2.19 1.7 0.00 1.58 1.32 1.7 0.3 0.78 0.30 2.86 0.00 0.00
2      B 465 164    W 1.54 1.8 0.78 1.83 0.95 1.3 2.2 1.53 1.56 1.93 0.00 0.00
3      C 480 143    W 0.78 1.4 0.78 1.49 0.48 1.3 1.1 1.83 2.33 1.87 0.00 0.48
4      D 418 107    W 0.00 1.6 1.75 0.78 0.60 1.8 0.9 1.38 0.85 0.00 0.00 0.00
5      E 306 138    P 1.51 2.0 0.85 1.11 1.38 1.8 1.0 2.61 1.77 0.00 0.48 1.15
6      F 282  45    F 0.30 1.6 0.30 1.90 1.83 1.6 1.9 1.56 0.85 0.00 0.00 0.00
7      G 263 284    F 0.78 1.2 0.48 1.45 2.16 1.7 1.4 0.90 0.90 0.48 0.00 1.18
8      H 520 280    P 0.60 1.3 0.00 2.50 1.90 1.3 1.3 0.70 1.69 0.90 0.00 0.00
9      I 575 266    F 0.00 0.9 0.00 2.10 1.58 1.0 2.0 1.92 0.60 0.00 0.00 0.00
10     J 363 594    W 1.81 2.6 0.30 0.00 1.68 0.0 0.0 0.00 0.00 1.90 1.82 0.00
11     K 260 756    P 1.43 2.6 0.48 0.00 2.36 0.0 0.0 0.00 0.00 2.11 2.60 0.00
12     L 237 809    W 0.95 2.5 1.98 0.48 2.04 0.0 0.0 0.90 0.00 0.00 1.75 0.00
13     M 316 864    W 0.85 2.2 0.00 0.85 1.59 0.0 1.0 0.00 0.00 0.00 1.04 0.00
14     N 263 874    W 1.00 2.7 0.00 0.00 2.20 0.0 0.0 0.00 0.00 2.63 2.15 0.85
```

Focusing on sites K and L only again.

```
round(mothslog[mothspa$Site %in% c("K", "L"), 5:16], 2)
```

```
     M1   M2   M3   M4   M5 M6 M7  M8 M9  M10  M11 M12
11 1.43 2.57 0.48 0.00 2.36  0  0 0.0  0 2.11 2.60   0
12 0.95 2.49 1.98 0.48 2.04  0  0 0.9  0 0.00 1.75   0
```

With numeric data distances rather than similarities are commonly used. For example, the squared (and normalised) Euclidean distance between K and L is

$$d^2_{K,L} = [(1.43 - 0.95)^2/(2.19)^2 + (2.57 - 2.49)^2/(1.83)^2 + \cdots]/12$$

where 2.19 and 1.83 are the ranges (max − min) across all 14 sites of the log counts for species 1 and 2.

This particular version of the Euclidean distance forces the distance to lie in the range $[0, 1]$, but the exact formula used under the heading *Euclidean* will be software package specific. It is common for Euclidean coefficients not to divide each term by its range, for example choosing `method = "euclidean"` in the `dist` function computes it using the original definition. This stresses again the need to check the details of the function used.

Deriving normalised squared Euclidean distances, as defined above, within `vegan` is awkward. Before calling `vegdist` the variables have to be normalised; that is, divided by their range. The derived distances then have to be squared and divided by the number of variables.

```
stmothslog <- decostand(mothslog[5:16], method = "range")
round(stmothslog, 2)
```

```
     M1   M2   M3   M4   M5   M6   M7   M8   M9  M10  M11  M12
1  1.00 0.45 0.00 0.63 0.45 0.97 0.14 0.30 0.13 1.00 0.00 0.00
2  0.70 0.47 0.39 0.73 0.25 0.74 1.00 0.59 0.67 0.67 0.00 0.00
3  0.36 0.29 0.39 0.60 0.00 0.73 0.52 0.70 1.00 0.65 0.00 0.41
4  0.00 0.37 0.88 0.31 0.07 1.00 0.42 0.53 0.36 0.00 0.00 0.00
5  0.69 0.62 0.43 0.45 0.48 0.97 0.46 1.00 0.76 0.00 0.18 0.97
6  0.14 0.39 0.15 0.76 0.72 0.86 0.88 0.60 0.36 0.00 0.00 0.00
7  0.36 0.15 0.24 0.58 0.89 0.96 0.65 0.35 0.39 0.17 0.00 1.00
8  0.27 0.19 0.00 1.00 0.76 0.72 0.62 0.27 0.72 0.32 0.00 0.00
9  0.00 0.00 0.00 0.84 0.59 0.58 0.93 0.74 0.26 0.00 0.00 0.00
10 0.83 0.92 0.15 0.00 0.64 0.00 0.00 0.00 0.00 0.67 0.70 0.00
11 0.65 0.91 0.24 0.00 1.00 0.00 0.00 0.00 0.00 0.74 1.00 0.00
12 0.44 0.87 1.00 0.19 0.83 0.00 0.00 0.35 0.00 0.00 0.67 0.00
13 0.39 0.68 0.00 0.34 0.59 0.00 0.46 0.00 0.00 0.00 0.40 0.00
14 0.46 1.00 0.00 0.00 0.91 0.00 0.00 0.00 0.00 0.92 0.83 0.72
```

```r
eucldist2 <- vegdist(stmothslog[, 1:12], method = "euclidean") ^ 2 / 12
round(eucldist2, 2)
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.13
3  0.18 0.06
4  0.27 0.16 0.13
5  0.28 0.18 0.12 0.19
6  0.21 0.10 0.15 0.12 0.17
7  0.23 0.19 0.17 0.21 0.10 0.11
8  0.16 0.09 0.12 0.19 0.22 0.05 0.12
9  0.27 0.14 0.18 0.16 0.24 0.03 0.15 0.07
10 0.20 0.32 0.37 0.37 0.40 0.36 0.38 0.33 0.42
11 0.27 0.39 0.44 0.44 0.46 0.40 0.41 0.37 0.46 0.02
12 0.36 0.35 0.38 0.24 0.33 0.29 0.35 0.34 0.36 0.13 0.12
13 0.24 0.23 0.29 0.24 0.32 0.16 0.24 0.18 0.18 0.09 0.13 0.13
14 0.29 0.42 0.42 0.49 0.40 0.43 0.34 0.40 0.49 0.07 0.06 0.21 0.17
```

A drawback of the Euclidean distance is that it is sensible to even a few large data values (outliers). A popular robust alternative is the (normalised) *Manhattan* or *city-block* distance.

$$d_{K,L} = [|1.43 - 0.95|/(2.19) + |2.57 - 2.49|/(1.83) + \cdots]/12$$

Another known alternative is the *ecological* distance, which is the same as city-block but ignoring double zeros

$$d_{K,L} = [|1.43 - 0.95|/(2.19) + |2.57 - 2.49|/(1.83) + \cdots]/8$$

The final divisor is now 8 rather than 12 because there are 4 double zeros: species 6, 7, 9 and 12. Therefore only 8 variables contribute to the distance calculation. The normalised Manhattan and ecological distances can be obtained with `vegan` as follows.

```r
# Normalised Manhattan distance matrix in vegan package
manhdist <- vegdist(mothslog[, 5:16], method = "gower")
round(manhdist, 2)
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.27
3  0.36 0.19
4  0.37 0.31 0.29
5  0.41 0.32 0.30 0.34
6  0.32 0.22 0.31 0.22 0.34
7  0.36 0.35 0.30 0.33 0.26 0.22
8  0.31 0.25 0.28 0.34 0.40 0.18 0.23
9  0.38 0.29 0.36 0.30 0.42 0.12 0.31 0.21
10 0.35 0.47 0.55 0.56 0.56 0.51 0.56 0.49 0.56
11 0.42 0.52 0.59 0.59 0.60 0.55 0.56 0.53 0.61 0.08
12 0.48 0.53 0.57 0.38 0.50 0.44 0.49 0.49 0.50 0.23 0.23
13 0.37 0.44 0.47 0.37 0.41 0.32 0.40 0.33 0.34 0.22 0.27 0.23
14 0.45 0.61 0.59 0.66 0.58 0.60 0.50 0.55 0.63 0.16 0.14 0.30 0.30
```

```r
# Ecological distance matrix using the vegan package
# altGower omits double zeros and decostand() standardises
ecoldist <- vegdist(decostand(mothslog[5:16], "range"), method = "altGower")
round(ecoldist, 2)
```

```
      1    2    3    4    5    6    7    8    9   10   11   12   13
2  0.32
3  0.39 0.21
4  0.44 0.38 0.32
5  0.41 0.32 0.30 0.37
6  0.38 0.26 0.34 0.29 0.37
7  0.39 0.38 0.33 0.36 0.26 0.24
8  0.41 0.30 0.30 0.41 0.40 0.22 0.25
9  0.50 0.35 0.39 0.44 0.46 0.16 0.33 0.28
10 0.38 0.51 0.55 0.61 0.56 0.56 0.56 0.54 0.61
11 0.46 0.56 0.59 0.65 0.60 0.60 0.56 0.58 0.66 0.17
12 0.52 0.58 0.57 0.46 0.54 0.53 0.49 0.53 0.60 0.34 0.35
13 0.44 0.48 0.47 0.45 0.45 0.39 0.40 0.40 0.45 0.33 0.41 0.34
14 0.49 0.61 0.59 0.66 0.58 0.60 0.50 0.60 0.69 0.28 0.24 0.40 0.45
```

There are, again, many ways of calculating distances for numeric data and there is a lack of consistency across software platforms with respect to the details of these calculations and to their naming conventions. Other specialised R packages that implement a range of distance measures include `cluster`, `proxy` and `simba`.

5.5 Exercises

1. RAPD data. Derive a similarity matrix for these 25 nematodes species based on the RAPD marker data. These markers are binary variables. Does it make much difference whether you use simple matching or Jaccard? Does a visual inspection of the matrix in any way support the known species classifications of these samples?
2. SOUPS data. How similar are these 8 soups with respect to (1) Odour (2) Flavour? Does either of these similarity patterns match the texture assessments?

6 Multidimensional scaling

6.1 Introduction

We have noted that principal components analysis produces a low-dimensional representation of the data, typically in two dimensions to facilitate interpretation. The relative position of the observations or samples in this representation approximates the similarity between them according to the measured variables. For ordinary numerical data this spatial configuration approximates the Euclidean distance between the original samples. This chapter introduces a class of methods that aim to produce an analogous low-dimensional representation from a distance or similarity matrix.

This provides a more general and fit-for-purpose approach to undergo analysis of similarities between samples as it allows to:

- Use measures other than the Euclidean distance, actually whichever is regarded suitable for the nature of the data.
- Work directly with a distance or similarity matrix, i.e. one obtained as raw data and not one derived from an ordinary data set $\mathbf{X}$ consisting of observations for a number variables. For example, data obtained from judges assessing how alike a number of cars, wines, tissue lesions, etc. are.

Note that if working with numerical data and using the Euclidean distance then the configuration of points will coincide with that obtained using PCA. This stresses the idea that many multivariate methods are not exclusive, they can be used in conjunction, offering different insight into the same data from different viewpoints. These methods are known as *ordination* methods in ecological and environmental sciences, in reference to ordering samples along axes of scientific meaning, but they are also widely used in other disciplines,

Starting with a distance or similarity matrix, the aim is to find a small number of new axes, or dimensions, which explain most of the distance or similarity matrix information. We then plot the samples along these new axes and observe their new orderings or ordinations.

An intuitively appealing example of this process uses, as its starting point, the inter-town distances table often included in old-fashioned road atlases. Treating this as a distance matrix, we generate two axes from it such that the towns (samples), when plotted on these new axes, are approximately the same distances apart as those given in the original matrix. In this case, because the original distances are almost Euclidean, the first two new axes will account for almost all the distance information and our "map" of the towns will be quite accurate. However, there is indeterminacy in terms of orientation. Our new map might be upside down or run East − West instead of West − East. The original matrix of distances contains no information on this point so neither do the new ordinations.

The main output is a low-dimensional graphical representation where the spatial distribution of the points optimally approximates the relative similarity between the samples. There are several methods to produce this and they largely differ in how agreement between fitted distances and observed proximities is assessed. We focuses here on classical or metric *multidimensional scaling* (MDS), also known as *principal coordinates analysis*.

6.2 Basic elements

Given a distance matrix d_{ij} for n samples the aim is to find the best $(n-1)$-column matrix of coordinates to describe the d_{ij}, with "best" here meaning that the (Euclidean) distances δ_{ij} derived from the matrix of coordinates minimise

$$\psi = \sum (d_{ij}^2 - \delta_{ij}^2)$$

Each column of the matrix form an axis or dimension providing the *scores* for each sample. It is desirable that the first 2-3 axes account for most of the distance information, so that we can obtain a succinct 2D or 3D graphical summary of the original (sometimes large) similarity/distance matrix and facilitate the interpretation of the results. As for the previous techniques, the axes are uncorrelated and can be ordered so that they successively account for decreasing fractions of the original information. Note that in MDS we only have scores and not loadings, we work directly with the distance matrix and have no information about the original variables.

For interpretation it is useful to plot the scores labelled by factors of interest or against other relevant variables.

6.3 MDS of the moths data set

The base R function implementing MDS is `cmdscale`. We use the normalised Manhattan distance matrix of the logged species counts created previously and compute $k = 2$ MDS dimensions or axes.

```
# From previous chapter:
moths <- read.table("moths.txt", h = T, stringsAsFactors = TRUE)
mothslog <- moths # Create a separate data set
mothslog[, 5:16] <- log10(mothslog[, 5:16] + 1) # Log-transform the counts
mothslog$Type <- as.factor(mothslog$Type) # To be used later
# Normalised Manhattan distance matrix using vegan package
library(vegan)
manhdist <- vegdist(mothslog[, 5:16], method = "gower")
```

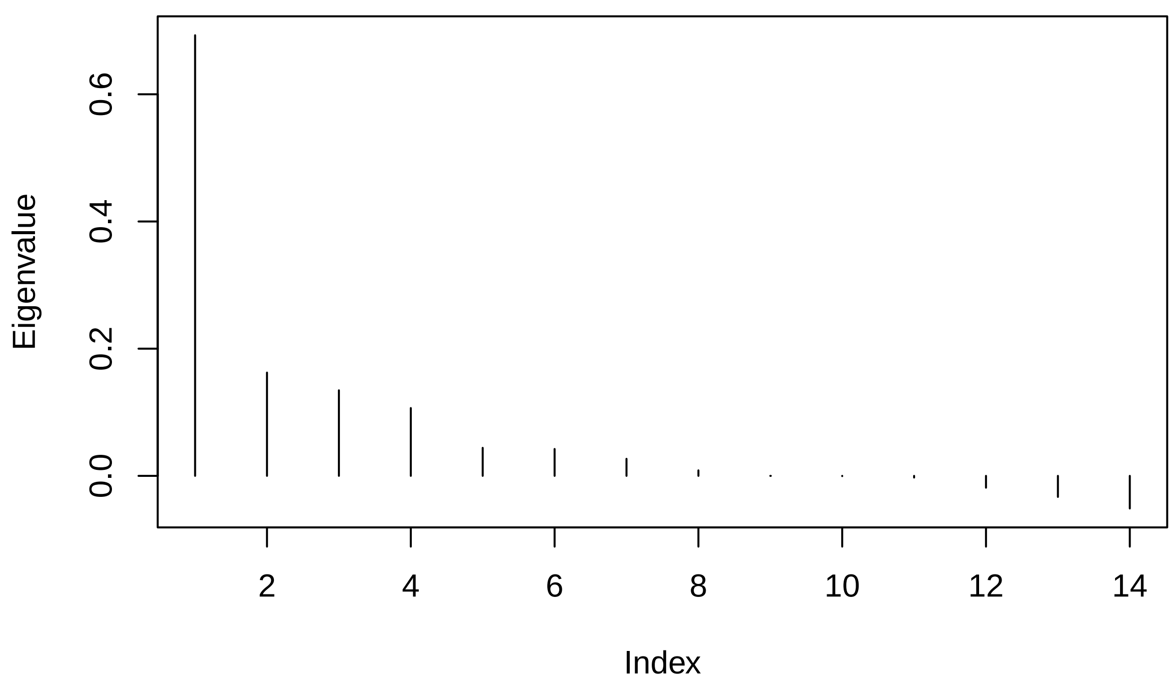
```
# Use Manhattan distance matrix as input for MDS
mds <- cmdscale(manhdist, k = 2, eig = T)
```

The results are held as follows:

- $points: coordinates for the MDS axes.
- $eig: contributions of each dimension to the total distance information provided by eigenvalues (similar to % variance due to each principal component in PCA).
- $GOF: goodness of fit, % distance accounted for by all k dimensions considered.

We can visualise the individual dimension contributions to represent the distances (eigenvalues, eig, are computed for all possible MDS axes, 14 in this case).

```
plot(mds$eig, type = "h", ylab = "Eigenvalue")
```



What is the goodness of fit (GOF) if we reduce to k dimensions? The value of the sum of the first k eigenvalues with respect to the total sum of eigenvalues could be used as for PCA, but this involves technical problems if no Euclidean distance is used and negative eigenvalues are produced. Two alternative measures are provided by the `cmdscale` function:

$$GOF = \frac{\sum_{i=1}^{k} |\lambda_i|}{\sum_{i=1}^{n} |\lambda_i|} \text{ and } GOF = \frac{\sum_{i=1}^{k} \max(0, \lambda_i)}{\sum_{i=1}^{n} \max(0, \lambda_i)}$$

Note that these two measures are equal when Euclidean distance is used.

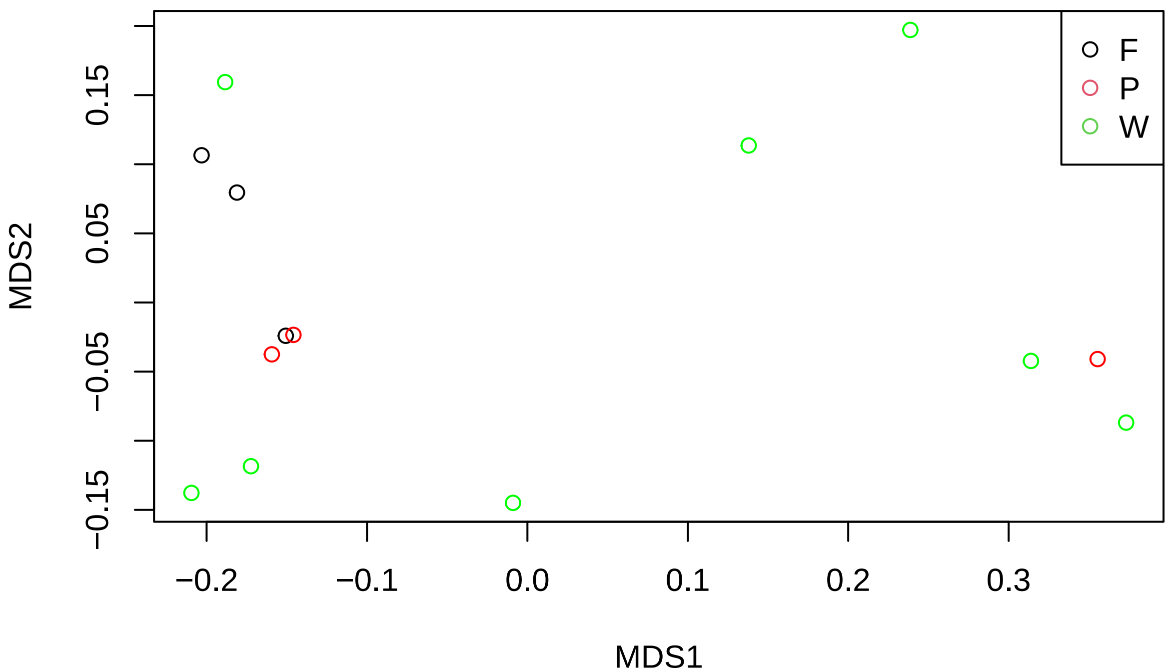
```
mds$GOF
```

```
[1] 0.64577 0.70204
```

The first of these measures suggests that 65% of the distance information is accounted for by the first 2 MDS axes. In fact, the plot above suggests that perhaps using up to four.

The following scatterplot shows the scores using the first two MDS axes. The points are labelled by site type.

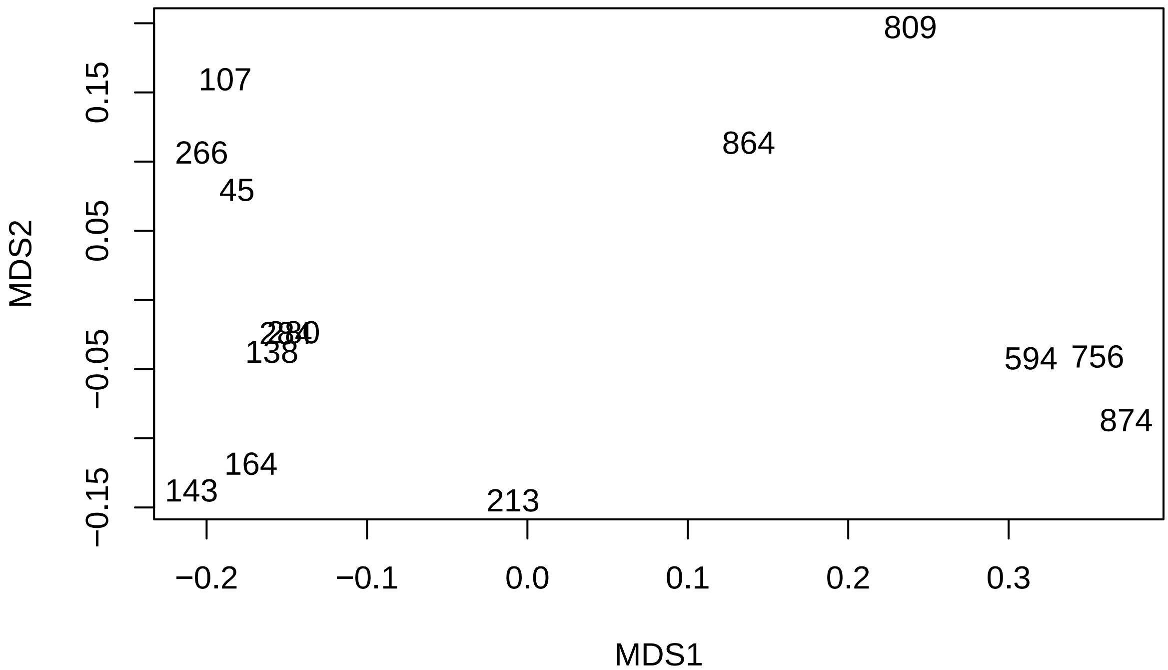
```
colours <- c("black", "red", "green", "blue")[mothslog$Type]
plot(mds$points[, 1:2], col = colours, xlab = "MDS1", ylab = "MDS2")
legend("topright", legend = levels(mothslog$Type), col = 1:3, pch = 1)
```



The plot above suggests that the 3 Farmland sites (in black, F) have similar moths profiles as they are close to each other, but there do not seem to be any other patterns.

However, we do have other variables in the data set, the Northings and Eastings and these can be easily incorporated into the graph.

```
# Plot scores with no symbol
plot(mds$points[, 1:2], type = "n", xlab = "MDS1", ylab = "MDS2")
# Use the Northings coordinates to label them instead
text(mds$points[, 1], mds$points[, 2], labels = mothslog$N)
```



This graph indicates that the first MDS dimension is mainly a North-South gradient.

6.4 Exercises

1. RAPD data. 26 nematode (tiny worm which lives in soil) samples representing 5 species were assessed on 122 RAPD bands (one of the early genetic marker technologies). Each band can take the value 1 or 0. Using the Jaccard coefficient derive a multidimensional scaling plot for the 25 samples. Do they separate into their known species? How many MDS dimensions are needed?
2. LIFEEXP data. Does a MDS analysis of these data provide a succinct summary of the similarities and differences among the countries with respect to life expectancy?

7 Linear regression

7.1 Introduction

In brief regression analysis is the study of the relationships between variables, typically one (or several) playing the role of response (dependent) variable and one or several playing the role of explanatory (independent) variables or predictors. We may want to do this for a number of reasons:

- We are interested in understanding the association between variables.
- We want to know how best to predict the value of a variable given the values of others, and how good this prediction is.
- We may want to investigate whether any relationship exists.

We are careful not to use words that imply that one variable causes changes in another. It may be so, but an association does not prove causation. Such things can only be established by careful interpretation of appropriately designed experiments, if this is possible.

Let us refer to the explanatory variable as X and the response variable as Y. We assume that X is known exactly. Some adjustment would need to be made to our results if it was believed to be subject to measurement error. We suppose that values of Y are random with a mean which is some function of X:

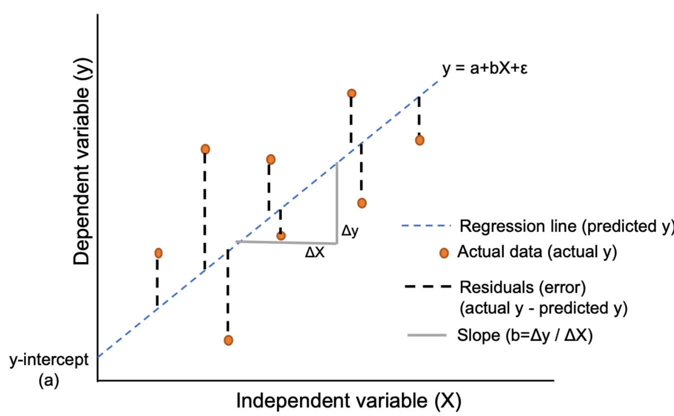
$$Y = f(X) + \epsilon$$

where ϵ is a random variable independent of X and usually assumed to have a $N(0, \sigma^2)$ distribution. For linear regression, we assume that $f(X)$ is a linear function, and so

$$Y = a + bX + \epsilon$$

Here the parameter a is the intercept: the mean value of Y when X is zero. The parameter b is the slope of the relationship: the average amount by which Y changes when X increases by 1.

We don't know the true values of a and b, but regression enables us to find the values of a and b which best fit the data. The *least squares* solution finds the values of a and b which minimise the sum of squared *residuals*. The following figure summarises the basic elements of simple linear regression.

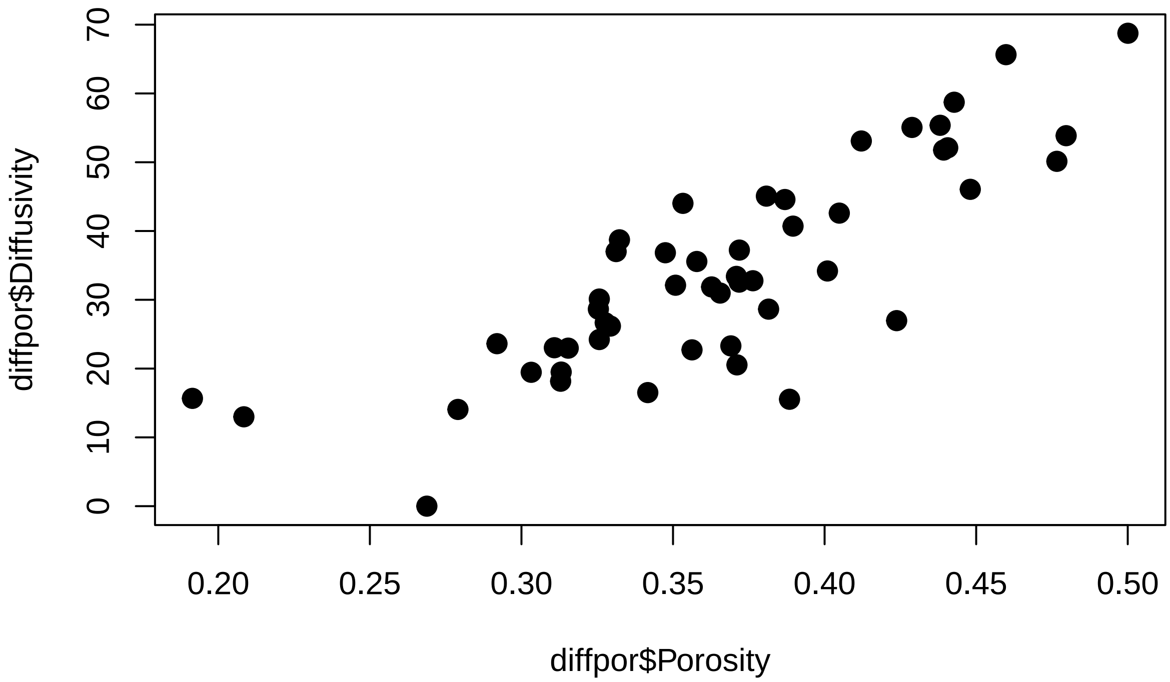


(https://www.reneshbedre.com/blog/linear-regression.html)

We illustrate these ideas by looking at some data on simulated soil diffusivity (`diffpor.txt`). The observations consist of 50 values of porosity and diffusivity and are plotted below.

We show below the output produced by R.

```r
diffpor <- read.table("diffpor.txt", header = TRUE)
# use plotting character 20 (filled circle), expanded by factor 2
plot(diffpor$Porosity,
     diffpor$Diffusivity,
     cex = 2,
     pch = 20)
```




```r
m.reg.1 <- lm(Diffusivity ~ Porosity, data = diffpor)
summary(m.reg.1)
```

```

Call:
lm(formula = Diffusivity ~ Porosity, data = diffpor)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.838  -3.652  -0.259   6.451  15.083 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -36.15       7.06   -5.12  5.3e-06 ***
Porosity      191.87      19.02   10.09  1.9e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.5 on 48 degrees of freedom
Multiple R-squared:  0.679, Adjusted R-squared:  0.673 
F-statistic:  102 on 1 and 48 DF,  p-value: 1.92e-13
```

There are several parts to the output:

- A description of the *model* being fitted.
- A summary of the distribution of the residuals.

- The regression coefficients and their standard errors. The coefficients are the values of a and b in the regression equation. The standard errors indicate how much they are subject to sampling variability (in other words they indicate the precision with which a and b have been estimated). The t-statistics are the ratios of the coefficient to their standard errors, and provide a way of assessing whether the coefficients differ significantly from zero. P-values are also given. Note that if the parameter $b = 0$, this implies no relationship between X and Y. If $a = 0$, then Y is directly proportional to X, and the line passes through the origin.
- Some summaries of goodness-of-fit. These indicate how well the regression model describes the data. In other words, how much of the variation in Y can be accounted for by variation in X, and how much remains.
  - Multiple $R^2 = 1 -$ (residual sum of squares)/(total sum of squares)
  - Adjusted $R^2 = 1 -$ (residual mean square)/(total mean square)

  both R^2 are expressed as a value in $[0, 1]$. The adjustment referred to has the effect of offsetting the tendency for R^2 to increase with additional explanatory variables in multiple regression (i.e. more than one X variable), even when they have no explanatory power. To obtain the sums of squares and mean squares we need to use the `aov` function (see below).
- Residual Standard Error is the estimate of σ, the standard deviation of the residuals. The σ quantifies how spread out around the regression line the data values are.
- F-Statistic refers to a test of whether the fitted model is a significant improvement over the mean of Y in predicting the individual Y values. In the case of simple linear regression the significance of the F-statistic is identical to the significance of t-test for $b = 0$.

In addition to using `lm` to obtain estimates of the regression line parameters, we can use `confint` to compute confidence intervals for the parameters.

```
confint(m.reg.1, level = .95)
```

```
              2.5 %  97.5 %
(Intercept) -50.343 -21.956
Porosity    153.618 230.120
```

7.2 Partitioning the data variation

```
summary(aov(m.reg.1))
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)    
Porosity     1   7345    7345     102 1.9e-13 ***
Residuals   48   3466      72                    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `aov` (analysis of variance) function partitions the total variation in the Y values, $\sum_{i=1}^{n}(Y_i - \bar{Y})^2$, into two components: the regression sum of squares (SS), $\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2$, and the residual sum of squares, $\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$. In a good model, one that describes the data well, the regression sum of squares will be relatively large and the residual sum of squares will be relatively small.

$$\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 + \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Associated with each sum of squares is the *degrees of freedom*, generally 1 fewer than the number of terms contributing to the sum of squares.

For the Total SS, there are n values, so df(total) $= n-1$. For the Regression SS there are two terms in the regression model, so df(reg) $= 2-1$ or 1. The Residual df is what is left after fitting the model, namely $(n-1)-(2-1)$ or $(n-2)$.

The term *mean squares* (MS) refers to the sum of squares divided by its degrees of freedom. Mean squares can be thought of as variances.

Mean Squares (Total) $= \sum_{i=1}^{n}(Y_i - \bar{Y})^2/(n-1) =$ variance (Y)

Mean Squares (Regression) $= \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2/1 =$ variance (regression)

Mean Squares (Residual) $= \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2/(n-2) =$ variance (residuals)

The F-statistic from the `lm` function and the F-Ratio from the `aov` function are the same thing, namely

$$F = \frac{MS(regression)}{MS(Residual)}$$

For a good regression model this ratio will be greater than 1.The larger this ratio, the better the model describes the data. The closer the F-ratio is to one, the less the variation about the regression line and the variation among the residuals are distinguishable.

The F can be thought of as a signal-to-noise ratio, and it is also the test statistic to test the hypotheses $H_0 : b = 0$ versus $H_A : b \neq 0$. Large positive values of F are evidence away from H_0 towards H_A.

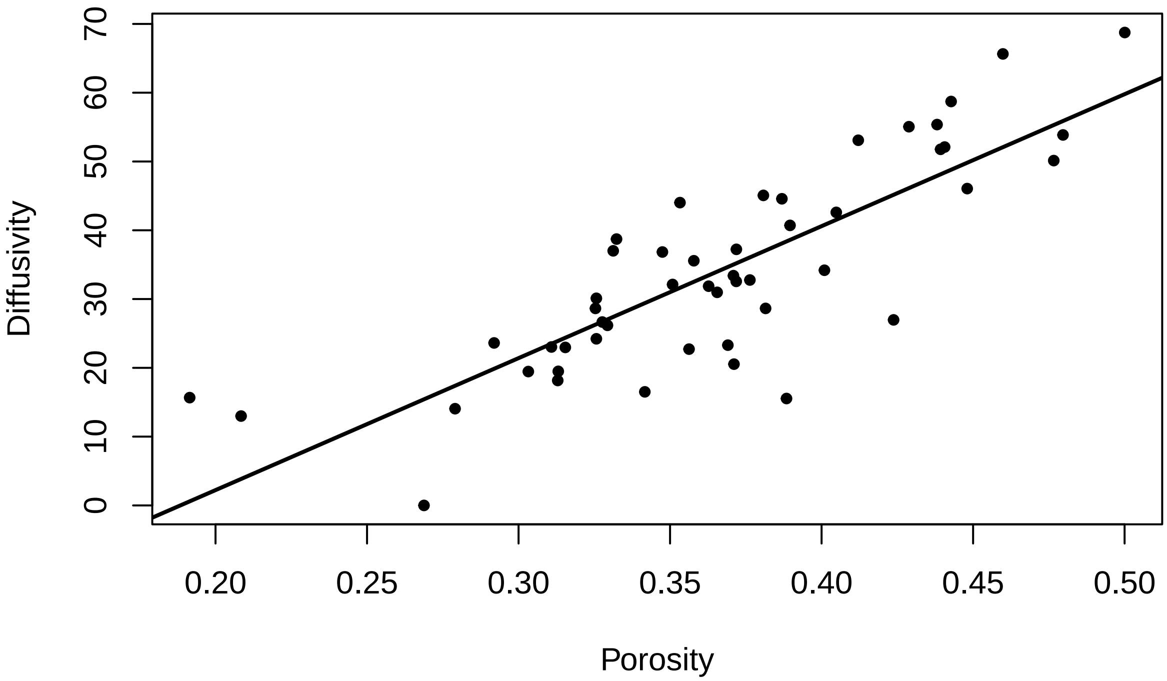
Under H_0 F has an F probability distribution. Associated with the F distribution are two parameters, the numerator (regression) degrees of freedom and the denominator (residual) degrees of freedom. In our case these parameter values are 1 and $(n-2) = 48$.

Features of the F distribution can be obtained in R in the usual way by the functions `rf`, `pf`, `qf` and `df`, analogous to the functions `rnorm`, `pnorm`, `qnorm` and `dnorm`.

7.3 Diagnostic plots

We can examine a plot including the fitted line. This should give us a good idea of how well the regression describes the data, or whether there are any problems with the fit.

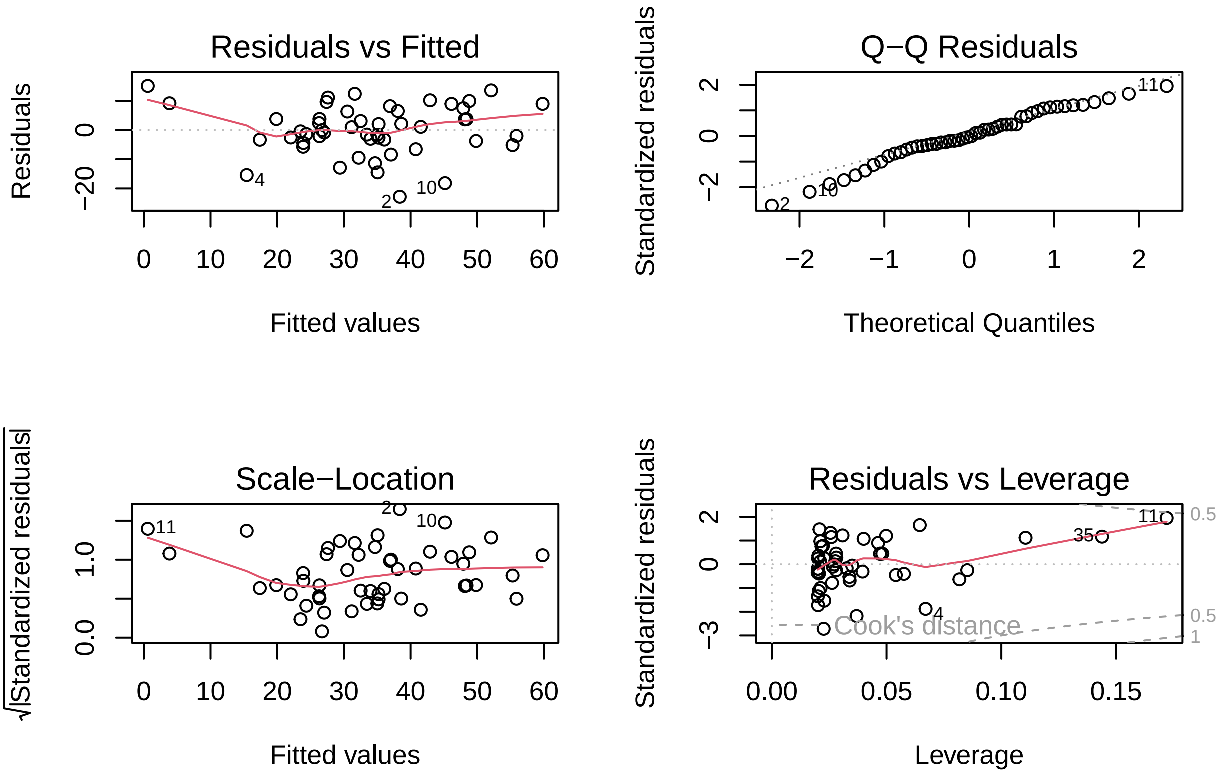
```
with(diffpor,plot(Porosity,Diffusivity,pch=20))
abline(m.reg.1,lwd=2) #add a line to current plot
```



However, *diagnostic* plots, based on the regression residuals, are designed to highlight the specific nature of any problems with the fitted regression model. They allow us to investigate whether each of the underlying assumptions of the regression model are justified. Specifically we assess whether the residuals have a constant variance and whether they are approximately normally distributed.

Common diagnostic plots based on residuals can be obtained by using `plot` on the `lm` object.

```
# Copy of current graphic parameters
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.reg.1)
```



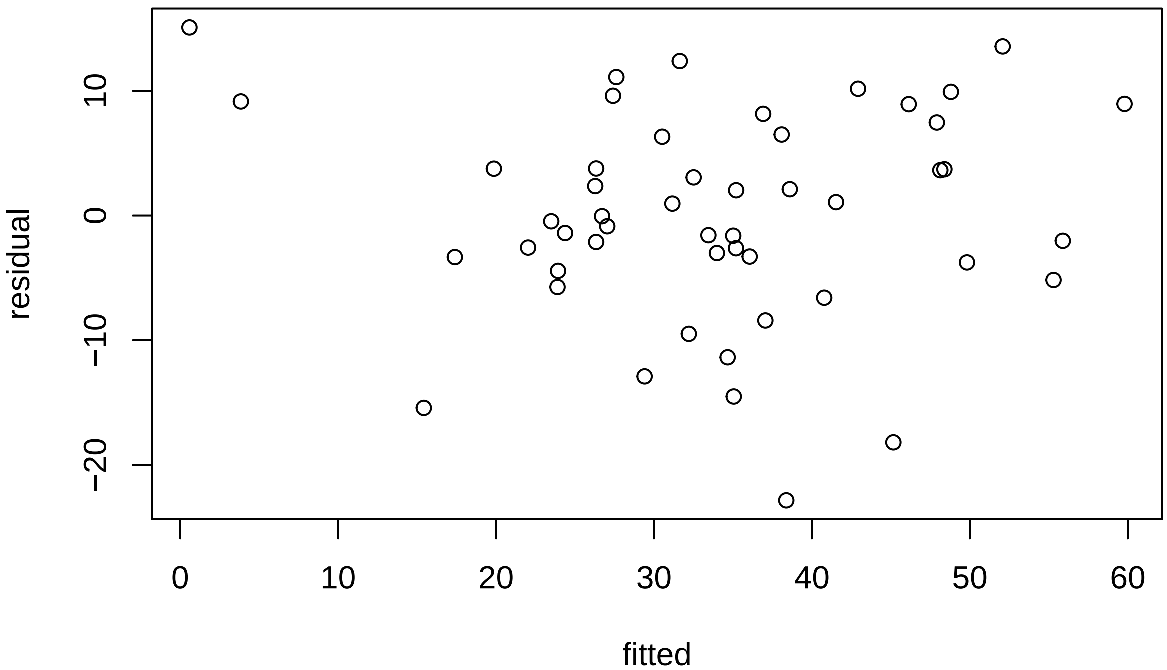

```r
# Revert to original settings
par(oldpar)
```

The residuals are the differences between the observations and their fitted values from the regression line. There should ideally be no pattern discernible in the residuals. If there were, it could be incorporated in the model. The best way to examine the residuals for pattern is to plot residuals against fitted values. This is shown in the top left plot. The only comment of note here seems to be that the residuals for the two lowest fitted values are positive, and by looking at the previous figure also, we may see a hint of curvature in the relationship. We might explore further by omitting these two observations and seeing how the fitted curve is affected (see below). Another pattern to look out for is residuals becoming more variable with increasing fitted value. This might indicate a need to transform X and/or Y.

It is assumed in much of linear regression that the residuals have a normal distribution. This can be checked by examining a q-q plot of residuals (top right graph). The departure from a straight line here is not too bad though the systematic deviation from the straight line for the lower porosity values is worth noting. Serious departure from normality might point to a need to transform the variables, or to fit a generalized linear model (GLM).

It is straightforward to get the residuals and fitted values explicitly from a regression.

```r
residual <- m.reg.1$residuals      #could also use resid(reg1)
fitted <- m.reg.1$fitted.values    #could also use fitted(reg1)
plot(fitted,residual)
```



There are other measures relevant to regression diagnostics. *Leverage* measures how far a point is from the centre of the independent (or explanatory) variable(s), that is the X variable(s). A point with high leverage need not necessarily indicate a problem, but it is always worth checking whether it is distorting the regression line. A point whose removal from the data set causes a large change in the regression equation is said to have high influence. *Cook's D* statistic is a popular measure of influence: a value greater than 1 suggests high influence.

7.4 Prediction

Sometimes the reason for deriving a regression line is to predict or estimate the value of the response variable at specific values of the independent variable. In general terms, given a new set of X values, we want to predict the corresponding Y values. These predictions are best accompanied by a confidence interval, to give an indication of the precision of our estimates.

We must distinguish between a confidence interval for the mean value of Y, and the confidence interval for a single value of Y, called the *prediction interval* in R. The estimate is the same in both cases, but the interval will be wider for the prediction of a single instance of Y rather for the mean value of Y. To understand the difference between these two situations consider the use of life expectancy tables. As an individual you are interested in finding the estimate of your own life expectancy and want a confidence interval for that single value estimate. From a life assurance company's perspective interest lies in average life expectancies and confidence intervals for estimates of these averages.

```
newvalues <- c(.25, .3, .35, .4, .45, .5)
newPorosity <- data.frame(Porosity = newvalues)
ci <- predict(m.reg.1, newPorosity, interval = "confidence")
pi <- predict(m.reg.1, newPorosity, interval = "prediction")

options.default <- options()     #save the default settings
options(digits = 3)      #restrict the number of digits printed
ci
```

```
   fit   lwr  upr
1 11.8  6.78 16.9
```

```
2 21.4 17.93 24.9
3 31.0 28.52 33.5
4 40.6 37.85 43.3
5 50.2 46.16 54.2
6 59.8 54.11 65.5
```

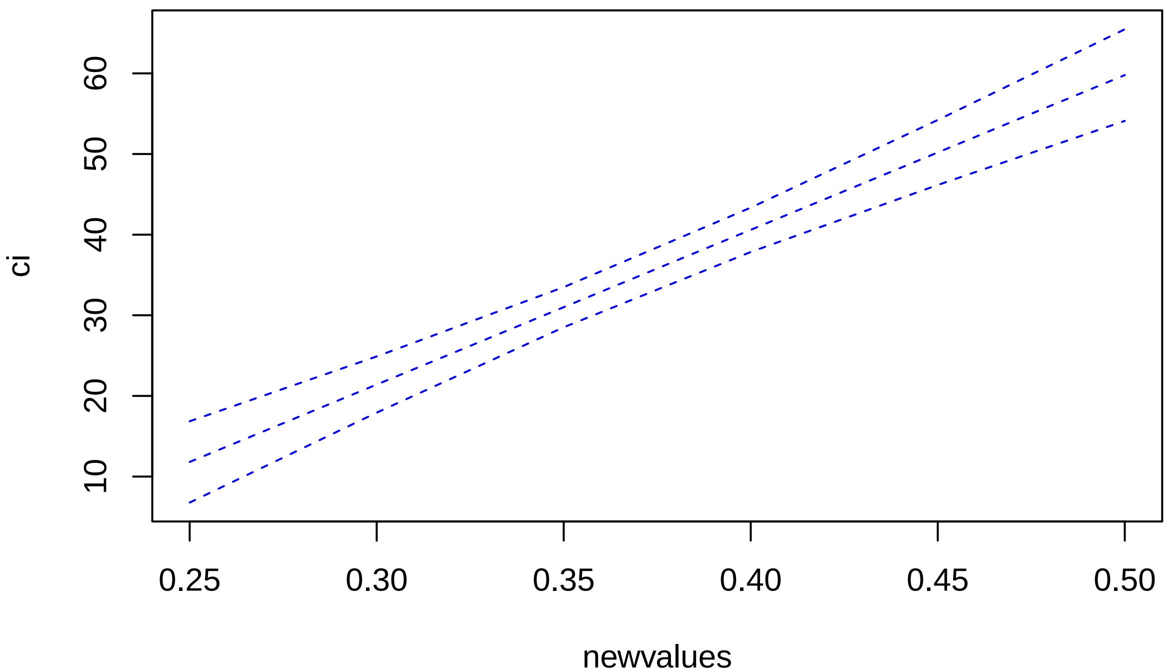
```r
pi
```

```
   fit   lwr  upr
1 11.8 -6.00 29.6
2 21.4  3.97 38.8
3 31.0 13.74 48.3
4 40.6 23.29 57.9
5 50.2 32.64 67.7
6 59.8 41.78 77.8
```

```r
options(options.default)     #reset to the default settings
```

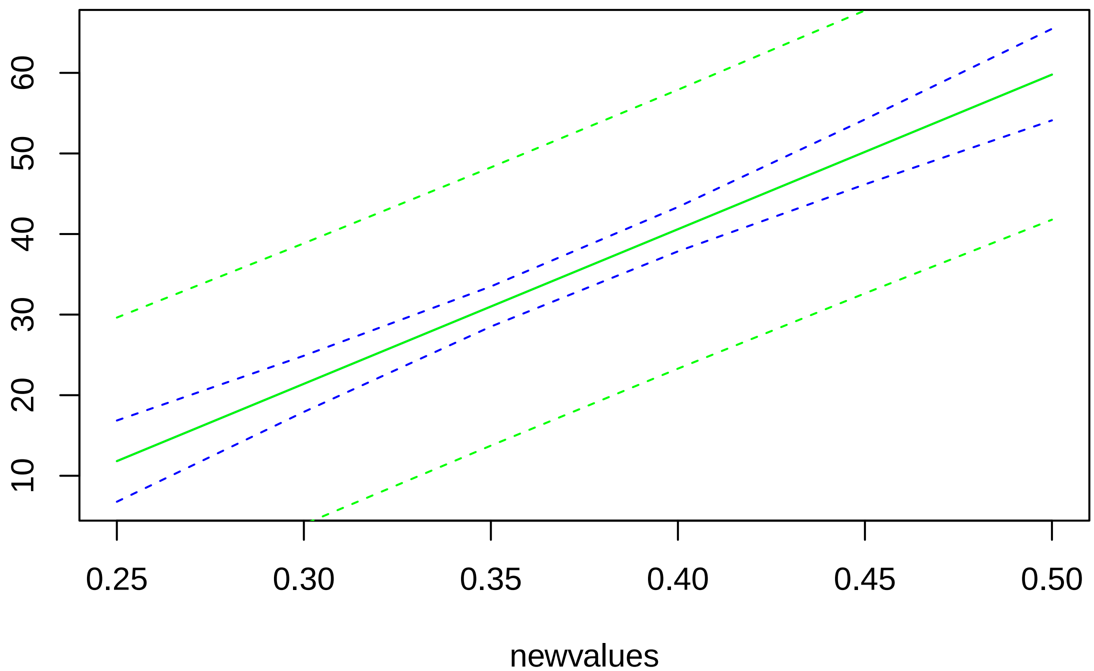
We can use the R function `matplot` (matrix plot) to plot several variables simultaneously. Here we plot all the columns of *ci* and *pi* against *newvalues*.

```r
# Confidence Interval for the mean value of Diffusivity
matplot(newvalues,
        ci,
        type = "l",
        col = "blue",
        lty = 2)
```




```r
# Confidence Intervals for the mean (blue)
# and predicted (green) values of Diffusivity
matplot(newvalues,
        ylab = "",
        ci,
        type = "l",
        col = "blue",
```

```
        lty = c(1, 2, 2))
matplot(
  newvalues,
  pi,
  type = "l",
  col = "green",
  lty = c(1, 2, 2),
  add = TRUE
)
```



7.5 Omitting specific data values

We mentioned above the possibility of investigating the effect of excluding the two observations with lowest porosity from the regression.

```
m.reg2 <- lm(Diffusivity~Porosity, subset=(Porosity>0.25), data=diffpor)
summary(m.reg2)
```

```
Call:
lm(formula = Diffusivity ~ Porosity, data = diffpor, subset = (Porosity >
    0.25))

Residuals:
    Min      1Q  Median      3Q     Max
-22.806  -3.766   0.866   5.756  13.466

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -47.74       8.09    -5.9  4.0e-07 ***
Porosity      221.64      21.48    10.3  1.5e-13 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.1 on 46 degrees of freedom
Multiple R-squared:  0.698, Adjusted R-squared:  0.692
F-statistic:  107 on 1 and 46 DF,  p-value: 1.48e-13
```

We see that the slope seems to be different from the estimate based on all the data, namely 192.0. However there is a large amount of uncertainty with these estimates ($se = 21$, approximately). The diagnostic plots (not shown) should play a bigger part in determining which the more appropriate model is. We note in passing that the goodness-of-fit statistics have moved in the direction of *improvement*: the residual standard error has decreased and adjusted R^2 has increased despite the use of two data points fewer. This supports our suspicions that the relationship may not be linear over the full range of porosities.

7.6 Transformation to linearity

Some other experiments have suggested that a relationship

$$\text{Diffusitivity} = a + b\text{Porosity}^2 + \epsilon$$

may be appropriate. We can fit this using linear regression, since the relationship is linear in porosity squared.

```r
diffpor$Porosity2 <- diffpor$Porosity^2
m.reg3 <- lm(Diffusivity~Porosity2, data=diffpor)
summary(m.reg3)
```

```

Call:
lm(formula = Diffusivity ~ Porosity2, data = diffpor)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.047  -4.728   0.529   4.655  13.511 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)    -3.37       3.66   -0.92     0.36    
Porosity2     271.44      25.22   10.76  2.2e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.12 on 48 degrees of freedom
Multiple R-squared:  0.707, Adjusted R-squared:  0.701
F-statistic:  116 on 1 and 48 DF,  p-value: 2.16e-14
```

We see that the fit has improved (slightly higher adjusted R^2 and slightly lower residual standard error).

A final point to note is that the non-significant constant is coherent with the subject matter theory, since it is expected that zero porosity requires zero diffusivity. We can force the constant to be zero in the model.

```r
m.reg4 <- lm(Diffusivity~0+Porosity2,data=diffpor)
summary(m.reg4)
```

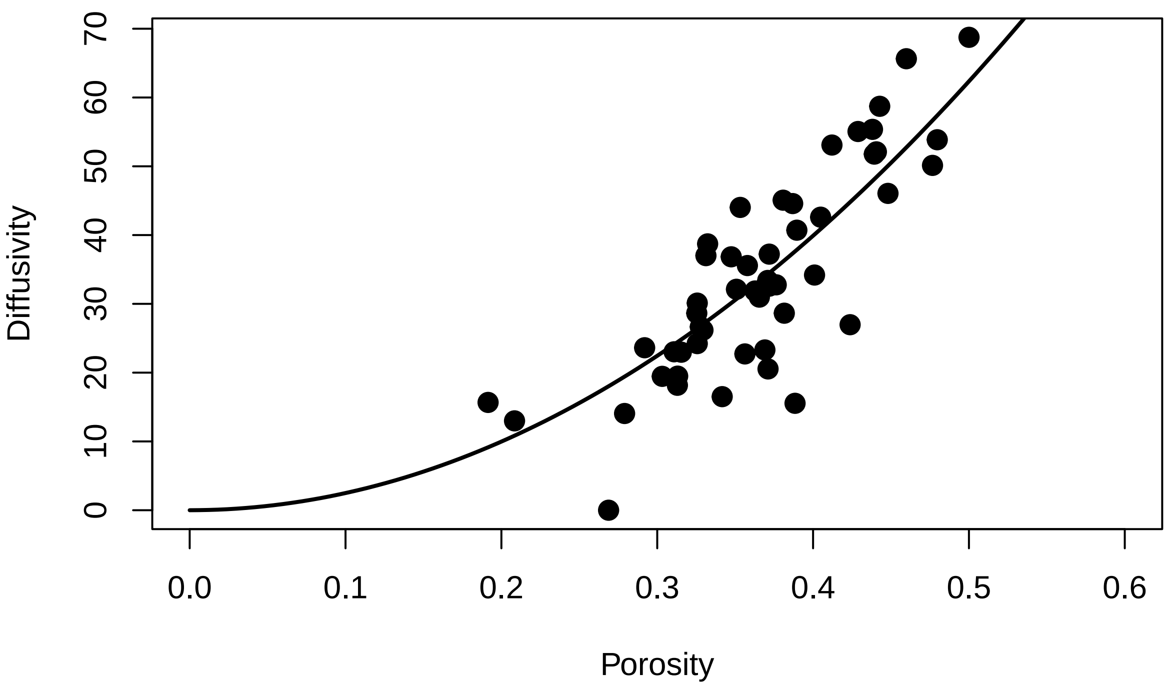
```

Call:
lm(formula = Diffusivity ~ 0 + Porosity2, data = diffpor)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-22.088  -4.722  -0.492   5.710  12.905 

Coefficients:
          Estimate Std. Error t value Pr(>|t|)    
Porosity2   249.40       7.91    31.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.11 on 49 degrees of freedom
Multiple R-squared:  0.953, Adjusted R-squared:  0.952 
F-statistic:  994 on 1 and 49 DF,  p-value: <2e-16
```

Note that the standard error of the slope (7.9) and the adjusted R^2 have both moved in the direction of *improvement*. Again the diagnostic plots (not shown) are more important and these show only a very modest improvement over the model Diffusivity $= a + b$Porosity $+ \epsilon$.

Note the in the figure below we have extended the fitted curve beyond the range of the data. This is purely to illustrate the fitted curve passing through the origin. In general one must be cautious to extend a model far beyond the range of the data unless there is good scientific justification for doing so.

```
with(diffpor,plot(Porosity, Diffusivity,  xlim=c(0,0.6),  pch=20, cex=2))
curve(249.4*x**2, from=0, to=0.6, add=T, lwd=2) # xlim=c(0,0.6) also works
```



However, the observation with third lowest porosity now looks even more to be too far below the curve. In the end we have to realise that these data alone cannot resolve the question of how the diffusivity/porosity relationship behaves below 0.3. Only more data could do that.

7.7 Exercises

1. With the DiffPorr data, multiply the Porosity values by 10 (Por10 = Porosity*10) and repeat the regression analysis from the notes. Which results change and by how much? Next add 5 to the

Diffusivity data (Diff5 = Diffusivity+5) and repeat the analysis from the notes. Which results change and by how much?

2. The file `leuchars.txt` has the minimum daily temperature in each month in Leuchars since 1957 (extracted from summaries available on a met office website). Investigate evidence for a linear trend in minimum daily temperature over time. Note that this file has 6 lines of introductory text. To skip this text use `read.table("leuchars.txt",skip=6,header=TRUE)`.

3. Look at the potato data, and investigate the association between length and weight (which is of interest to those who grade and process potatoes). Plot the association, try linear regression, interpret the results, examine the diagnostic plots. Would it be sensible to drop the intercept term from the regression? Derive 95% confidence intervals for the mean weight of a tuber for a sensible range of tuber lengths. How do these intervals compare with the variation in tuber weights? In other words have we gained much in predicting weight by using information about length?

4. Use the OX data to model the weight gain (Wtgain) of an animal as a function of its (a) current weight (Weight), (b) age (Age) and (c) food intake (Intake). Note that this data file uses asterisk to denote missing values. Either replace these by 'NA' in the text file, or, better, use `read.table("ox.txt",na.strings="*", header=TRUE)`. Investigate, compare and decide about a sensible regression model specification for this data set.

8 Nonlinear regression

In nonlinear regression we model an association where the response is not a linear function of the explanatory variable. Before considering nonlinear regression, it is always worth considering whether linear regression methods can be adapted, in either of two ways:

- Transform the explanatory and/or the response variable.
- Add additional functions (such as powers) of the explanatory variable, and perform multiple linear regression, the topic we examine in the following section.

The advantages of linear regression are that it is easier to do, more robust to fitting problems, and more widely available in statistical software. However, sometimes there are good reasons to fit a nonlinear function, or model, directly. The following examples show how to do this in R. They involve the `nls` function instead of `lm`.

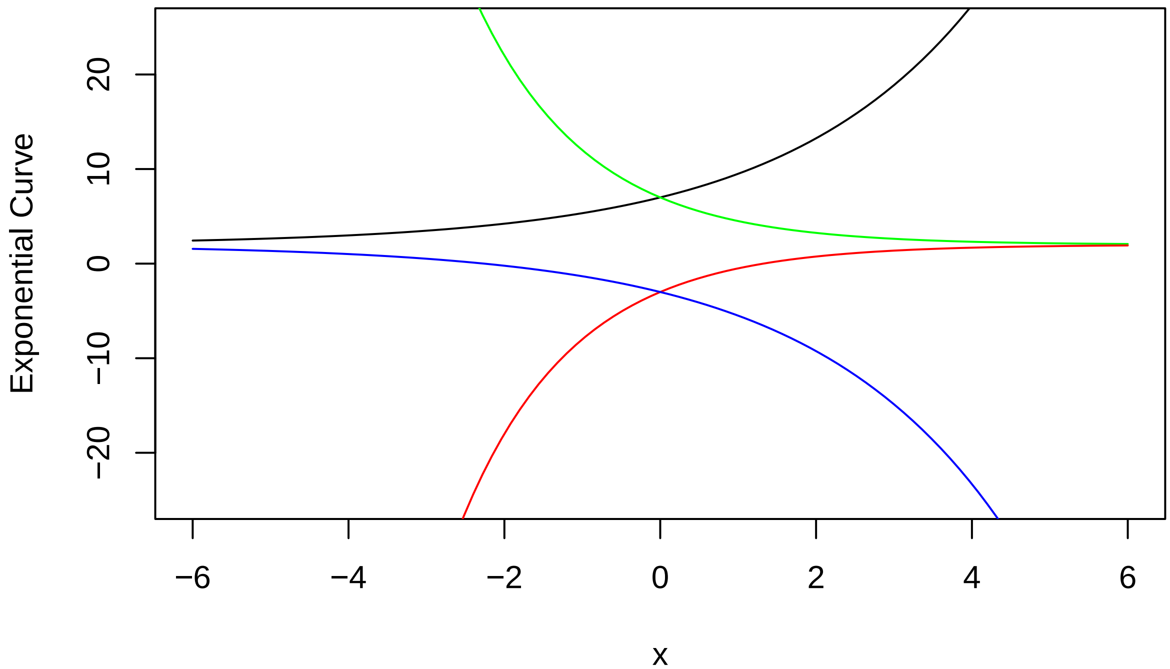
Fitting nonlinear curves requires having a feel for the form of the mathematical formula you are using. We consider here the two most common nonlinear shapes, exponential and sigmoid.

8.1 Exponential curves

Exponential relationships are frequently seen in e.g. the biological sciences, where they arise in situations where the rate of change in some variable Y is a linear function of itself. An example of this would be the growth of a colony of bacteria in some medium. Initially, the number of bacteria in each new generation would be proportional to the number in the preceding generation, and the colony will grow exponentially.

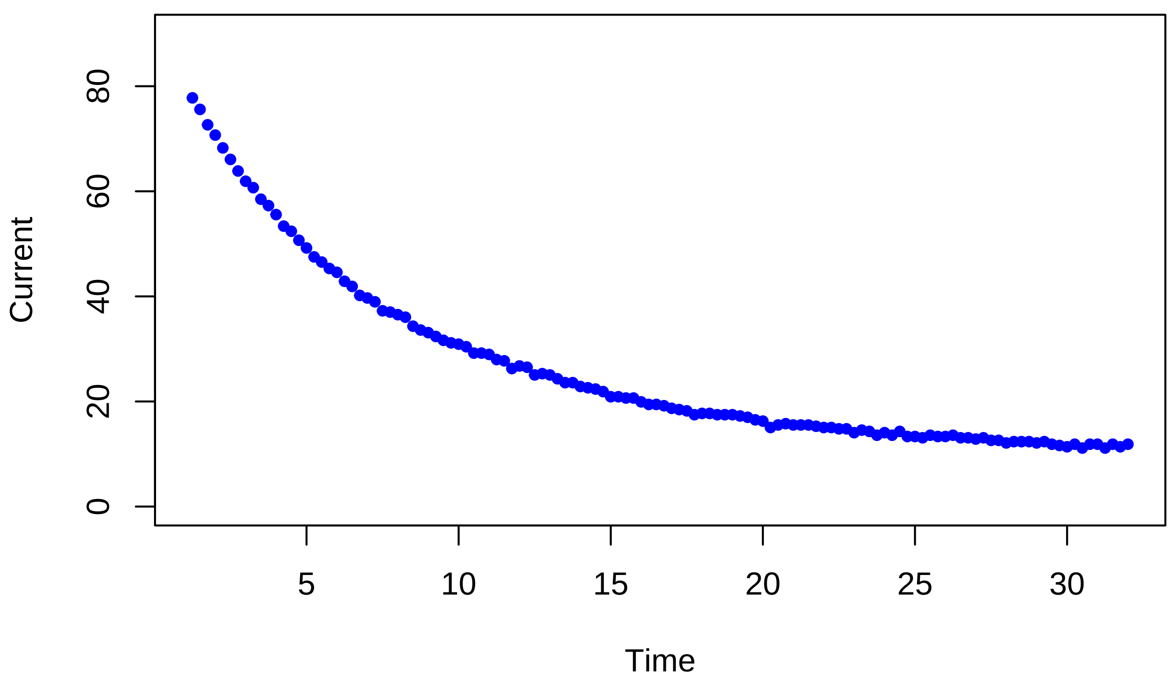
The standard exponential is defined by the relationship $Y = A + BR^X$. We have to estimate the three parameters A, B and R. The standard exponential is either continuously decreasing or increasing, and will be approximately either concave or convex everywhere in its range. When $X = 0$, $R^X = 1$ and therefore $Y = A + B$; as $X \to \infty$, Y either tends to an asymptote at A (if $0 < R < 1$) or tends to $\pm\infty$ (if $R > 1$). The shape of the curve depends on the values taken by B and R. The four possible scenarios are illustrated in the following figure showing some standard exponential curves of the form $Y = A + BR^X$ with $A = 2$.

```
x <- seq(-6,+6,.1)

curve(2+5*1.5^x, col='black',xlim=c(-6,6),ylim=c(-25,+25), ylab="Exponential Curve" )
curve(2+5*0.5^x, col='green',xlim=c(-6,6), ylim=c(-25,+25), add=TRUE )
curve(2-5*0.5^x, col='red',  xlim=c(-6,6), ylim=c(-25,+25), add=TRUE )
curve(2-5*1.5^x, col='blue', xlim=c(-6,6), ylim=c(-25,+25), add=TRUE )
```



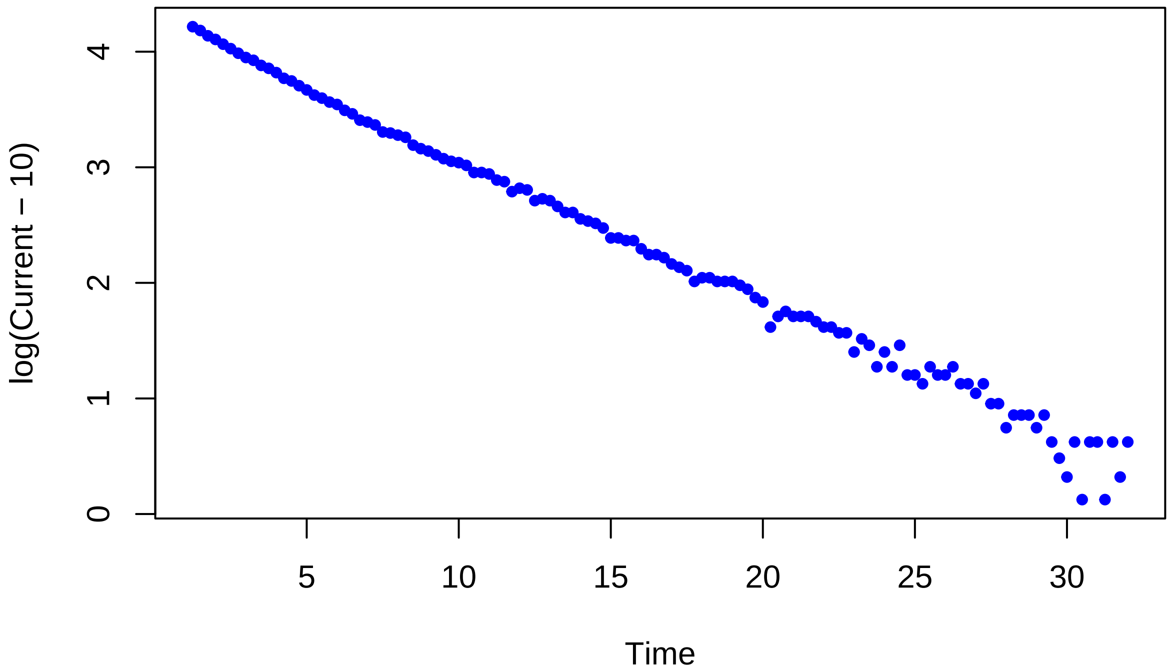
We will illustrate the use of the standard exponential using data from file `frog.txt`. Observations describe the electrical current flowing through the end-plate membrane of a frog's muscle fibre following a jump in voltage. We wish to examine how the current changes with time.

```
frog <- read.table("frog.txt", header=TRUE)
with(frog, plot(Time, Current, pch=20, col='blue', ylim=c(0,90)))
```



These data exhibit the characteristic shape of an exponential curve with $R < 1$ and $B > 0$. We can assess the suitability of the standard curve by guessing the asymptotic value A (≈ 10), subtracting it from the Y values, taking logarithms of the results and then plotting these values against *Time*. If the standard model is appropriate, the result should be a straight line.

```r
with(frog, plot(Time, log(Current-10), pch=20, col='blue'))
```



The linear nature of this plot suggests that the standard exponential is a reasonable function to try. We correspondingly fit the model $Y = A + BR^X + \epsilon$.

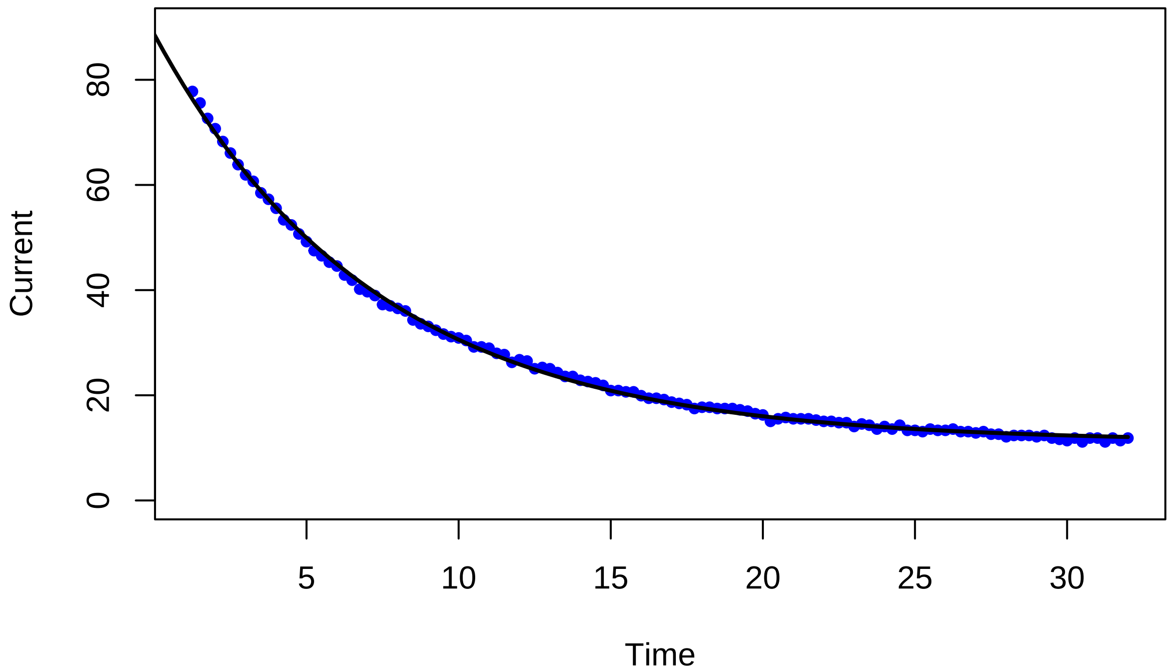
We can use the `nls` function for this, but it often requires starting values for the parameter estimates. Otherwise it will choose 1.0 by default, but it might not work and then we need to make some guesses. Using the following specifications will work here.

```r
m.frog.1 <- nls(Current ~ A+B*R^Time, start=list(A=10,B=1,R=0.5), data=frog)
summary(m.frog.1)
```

```

Formula: Current ~ A + B * R^Time

Parameters:
  Estimate Std. Error t value Pr(>|t|)
A 1.11e+01   1.21e-01    92.1   <2e-16 ***
B 7.74e+01   3.00e-01   257.8   <2e-16 ***
R 8.71e-01   9.57e-04   910.2   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.583 on 121 degrees of freedom

Number of iterations to convergence: 9
Achieved convergence tolerance: 2.29e-08
```

We use the parameter estimates to plot the estimated curve along with the observed data.

```r
with(frog, plot(Time,Current, pch=20, col='blue', ylim=c(0,90)))
curve(11.14 + 77.45*0.8709^x, 0, 32, add=TRUE, lwd=2)
```



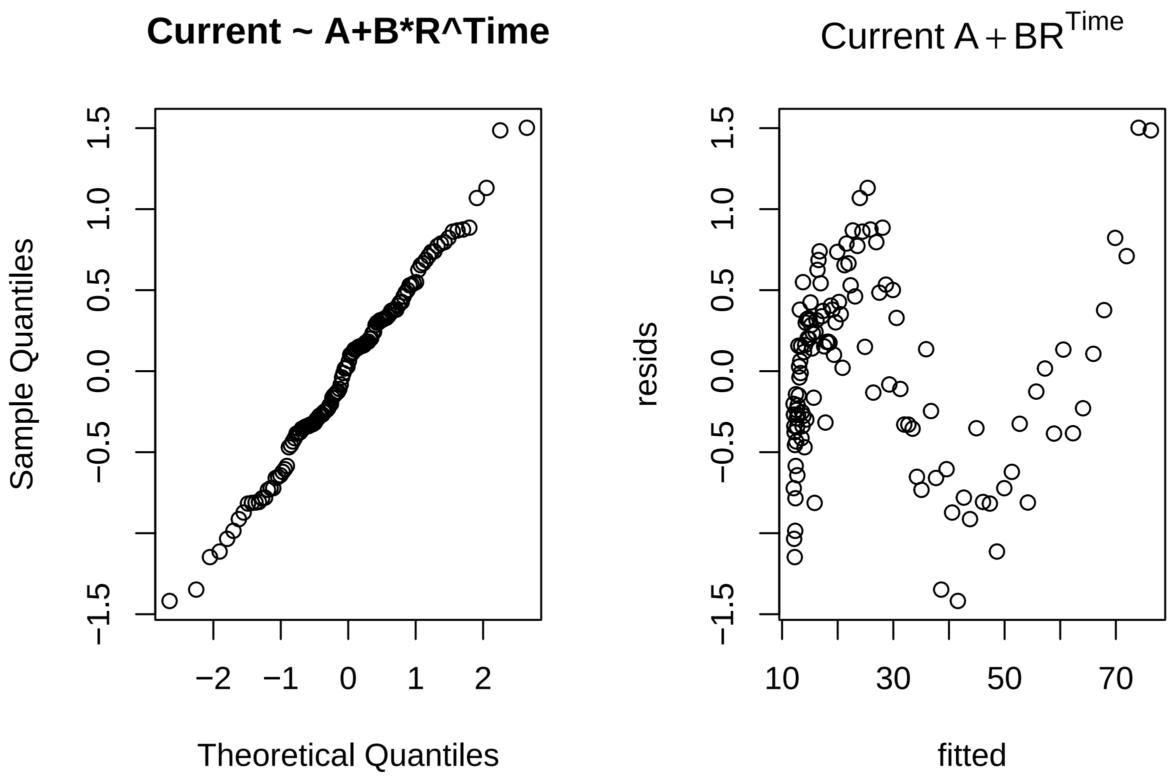
Confidence intervals for the parameter estimates are easy to obtain.

```
confint(m.frog.1)
```

```
      2.5%    97.5%
A 10.89231 11.37841
B 76.85114 78.05050
R  0.86897  0.87284
```

Diagnostic plots should always be examined, but these are not so conveniently obtained as in linear regression. Some manual computing can be use instead.

```
resids <- residuals(m.frog.1)
fitted <- fitted(m.frog.1)
par(mfrow=c(1,2))
qqnorm(resids, main="Current ~ A+B*R^Time")
plot(fitted, resids, main=expression(Current ~ A+B*R^Time))
```




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

The q-q plot looks good so the residuals overall are approximately normally distributed. However, there is clearly systematic deviation from the fitted model. When the fit of the data to the model is good as in this case (a very small residual sum of squares), it might be not worth the extra work finding out the cause of the systematic deviations.

Prediction using `predict` with `nls` models provides point estimates but not confidence intervals.

```
newTimes <- data.frame(Time=c(10.5,20.6,60.5))
predict(m.frog.1, newdata=newTimes)
```

```
[1] 29.281 15.629 11.155
```

8.2 Sigmoid curves

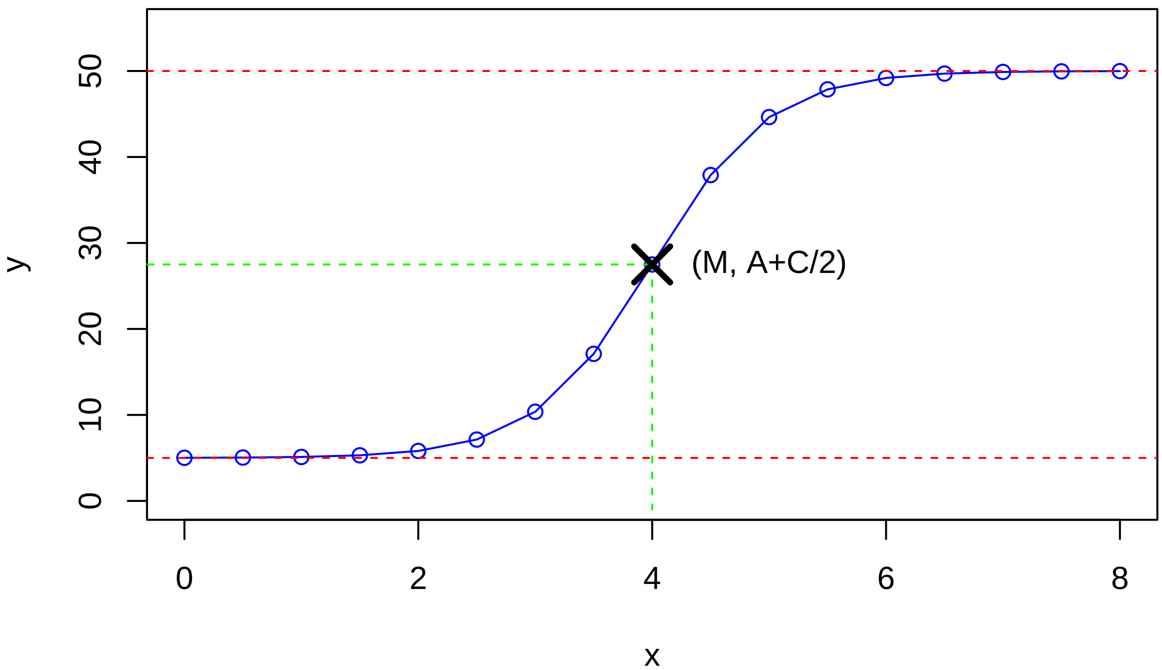
These are S shaped curves, most frequently use for *growth* models. They are monotonic (either increasing or decreasing) and approach asymptotes at both extremes of X. They will have a single point of inflexion, and hence will be concave in one section of their range and convex in the other. In general it is difficult to fit sigmoid curves accurately unless the full range of the curve (from lower asymptote to upper asymptote) is present in the data. An example would be the growth of bacteria in a closed medium. The population would initially grow exponentially, but as numbers became larger, the growth rate would increasingly become subject to constraints due to lack of resources. These constraints would become increasingly strong as the population grew larger, causing the growth rate to consistently decrease, and hence causing the population size to reach a plateau.

The **logistic function** defined by the relationship

$$Y = A + \frac{C}{1 + e^{-B(X-M)}}$$

specifies a symmetric sigmoid curve with a left-hand asymptote taking the value A, a right-hand asymptote taking the value $A + C$, and a central point of inflexion at $X = M$, $Y = A + \frac{C}{2}$, while the parameter B controls the rate at which the function moves between the two asymptotes: the greater B the steeper the slope. The following figure shows a standard logistic curve with parameters $A = 5, C = 45, B = 2$ and $M = 4$.

```
x <- seq(0,8,.5); A=5; C=45; B=2; M=4
y <- A+C/(1+exp(-B*(x-M)))
plot(x,y,col="blue", ylim=c(0,55))
lines(x,y,col="blue") # add line segments to join the points
abline(h=c(A,A+C),col="red", lty= "dashed")# add asymptotes
#next, indicate the point of inflection
lines(x=c(-1,M), y=c(A+C/2,A+C/2), col="green", lty="dashed")
lines(x=c(M,M), y=c(A+C/2,-1), col="green", lty="dashed")
# Add point and label at inflexion point
points(M,(A+C/2), pch=4,cex=2.5,lwd=3)
text(M+1,(A+C/2), "(M, A+C/2)")
```



The **generalized logistic (or Richard's) function** is defined by the relationship

$$Y = A + \frac{C}{(1 + Te^{-B(X-M)})^{\frac{1}{T}}}$$

This function specifies a five parameter asymmetric sigmoid curve. The parameters A, B, C and M have the same interpretations as for the logistic curve, but the parameter T, sometimes called the power law parameter, now specifies the value taken by the curve at the point of inflection, namely $Y = A + C/(1 + T)^{1/T}$. The asymmetry allowable in this sigmoid because of the parameter T provides greater flexibility in fitting to the data.

The **Gompertz function** is specified by the relationship

$$Y = A + Ce^{-e^{-B(M-X)}}$$

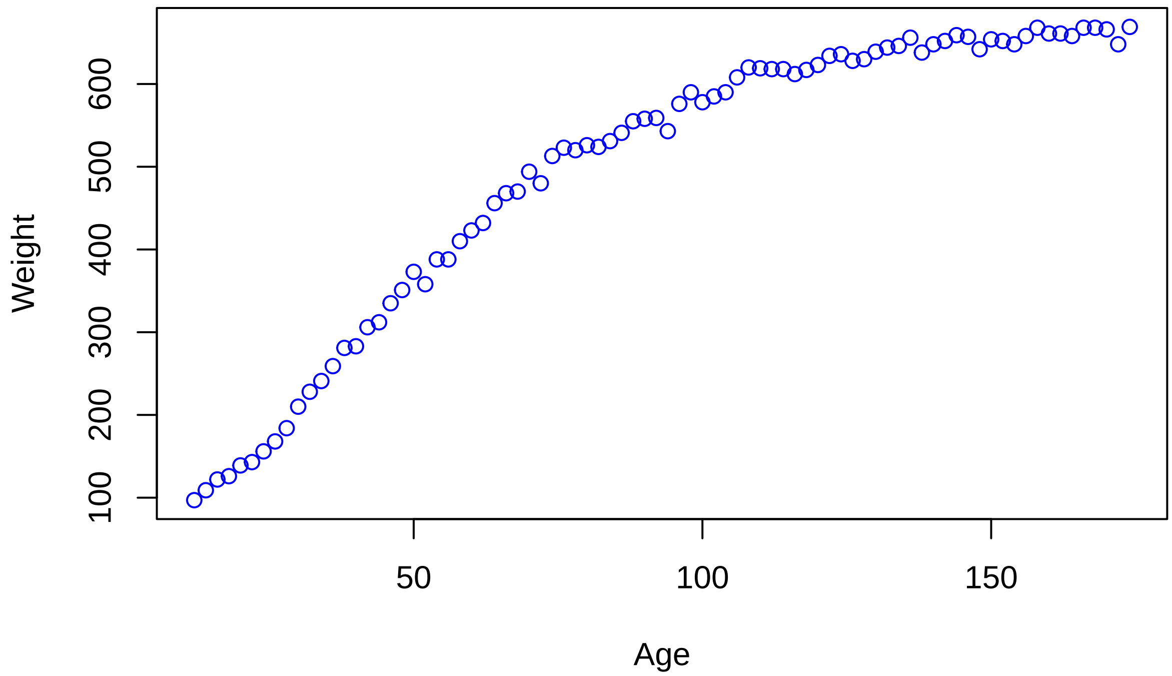
and defines a limiting case of the generalized logistic which occurs when $T \approx 0$. It is still asymmetric. The point of inflexion occurs at the point $X = M$ and the asymptotes are A and $A + C$.

These curves may be fitted to growth data of an ox (`ox.txt`). We plot the weight of the oxen against age. Fitting a logistic function here we can expect problems since the left-hand asymptote is not clearly visible in the dataset.

```
ox <- read.table("ox.txt", header=TRUE)
names(ox)
```

```
[1] "Age"    "Weight" "Wtgain" "Intake"
```

```
with(ox, plot(Age,Weight,col="blue"))
```



Fitting the logistic function:

```
m.ox.1 <- nls(Weight ~ A+C/(1+exp(-B*(Age-M))), start=list(A=0,C=600,M=50,B=0.05), data=ox)
summary(m.ox.1)
```

```


Formula: Weight ~ A + C/(1 + exp(-B * (Age - M)))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
A -2.06e+02   3.88e+01   -5.32  9.7e-07 ***
C  8.76e+02   4.08e+01   21.48  < 2e-16 ***
M  3.25e+01   2.63e+00   12.39  < 2e-16 ***
B  3.39e-02   1.23e-03   27.59  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.72 on 78 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 2.55e-06
```

```
confint(m.ox.1, level=.95)
```

```
         2.5%       97.5%
```

```
A -295.038326 -139.672768
C  805.800496  969.335898
M   26.706412   37.148479
B    0.031481    0.036366
```

Note the very wide confidence interval for each asymptote.

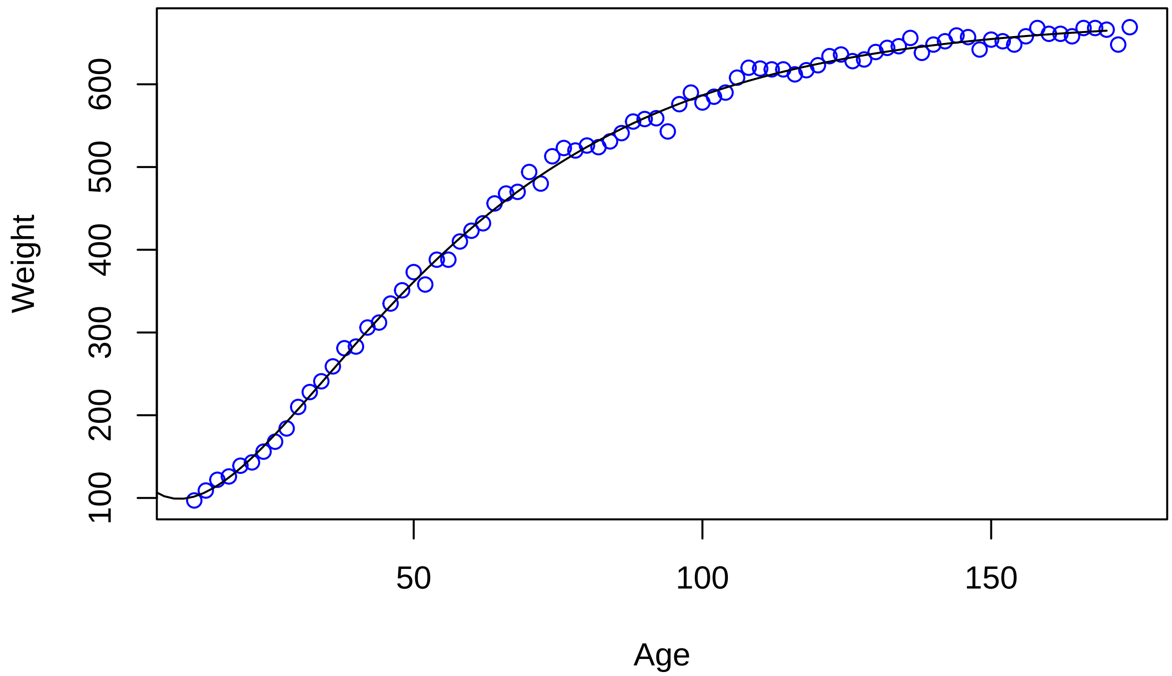
For the generalized logistic:

```
m.ox.2 <- nls(Weight ~ A+C/(1+T*exp(-B*(Age-M)))**(1/T),
                start = list(A=0,C=600,M=50,B=0.05,T=0.1),
                data=ox)
summary(m.ox.2)
```

```

Formula: Weight ~ A + C/(1 + T * exp(-B * (Age - M)))^(1/T)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
A  98.87437   13.43239    7.36  1.7e-10 ***
C 579.76119   12.38262   46.82  < 2e-16 ***
M  34.60807    1.32559   26.11  < 2e-16 ***
B   0.02753    0.00125   22.09  < 2e-16 ***
T  -0.50112    0.09707   -5.16  1.9e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.02 on 77 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 7.11e-06
```

We plot ox weight as a function of age with the generalized logistic curve.

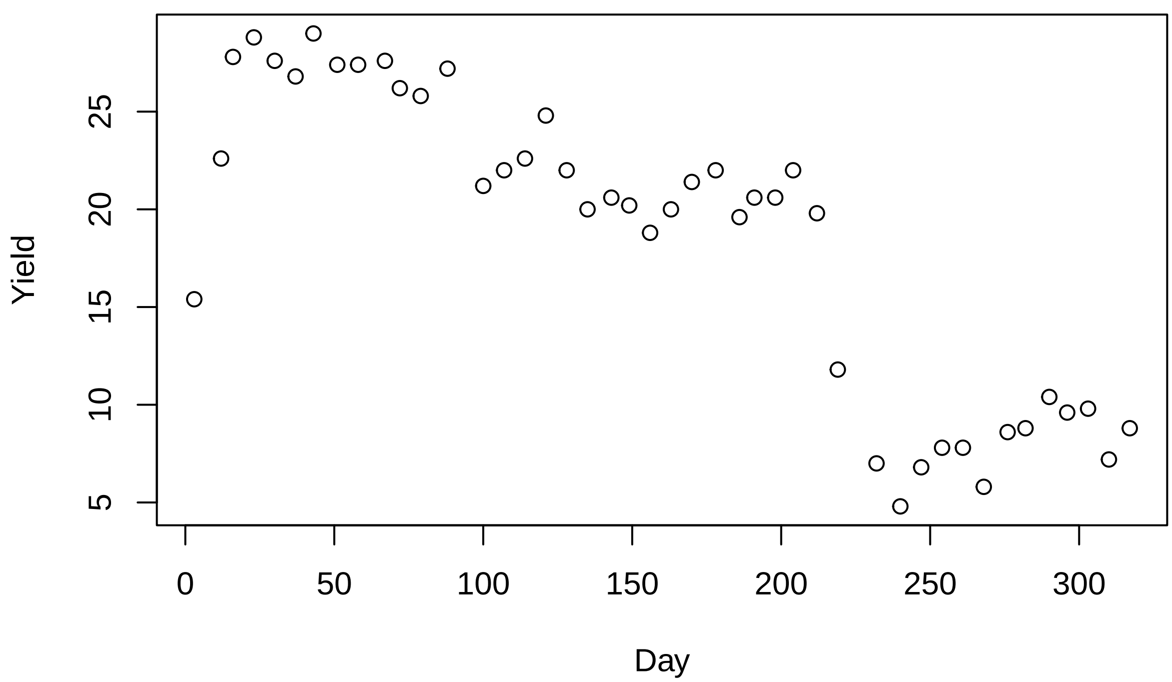
```
with(ox, plot(Age,Weight,col="blue"))
curve(98.9 + 579.8/(1 -0.5 * exp(-0.0275 * (x - 34.6)))^(1/(-
0.5)),0,170,add=TRUE)
```



8.3 Non-parametric curve fitting

We can also fit smooth curves without specifying a mathematical form for them. Plot the data in `lact.txt`. Its shape does not conform to either the logistic or exponential form. We want to relate milk yield with days birth.

```
lact <- read.table("lact.txt",header=TRUE)
with(lact,plot(Day,Yield))
```



The first two data points are not *bad values* but are exactly what was expected in this study. It is possible to generate a best-fitting *smooth spline* through the data points and use this fitted curve to predict milk yield for any new value of day.

From https://www.stat.berkeley.edu/~s133/Smooth1a.html:

Spline smoothing, named after a tool formerly used by draftsmen. A spline is a flexible piece of metal (usually lead) which could be used as a guide for drawing smooth curves. A set of points (known as knots) would be selected, and the spline would be held down at a particular x,y point, then bent to go through the next point, and so on. Due to the flexibility of the metal, this process would result in a smooth curve through the points

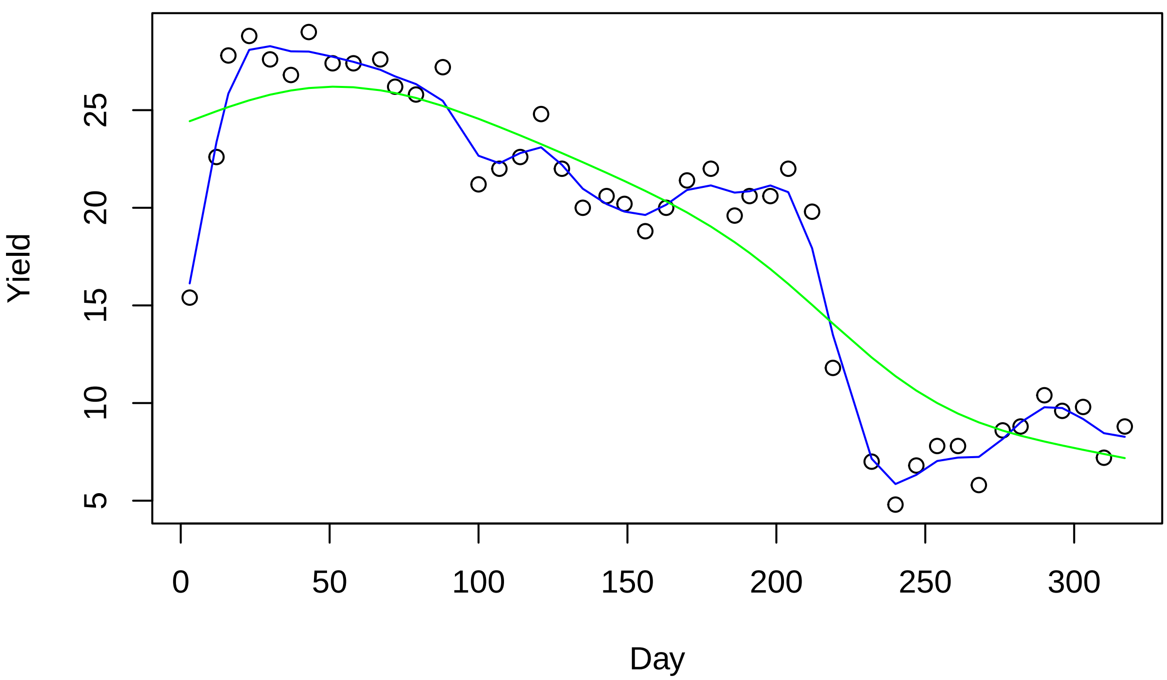
With statistical curve fitting *knot points* are chosen and a cubic curve fitted through all the data points between each consecutive pair of knot points, in such a way as to ensure smoothness at the joins. The smoothing parameter, `spar`, in effect determines how many knot points are to be chosen.

- `spar = 0` implies no smoothing, so a curve is fitted to every consecutive pair of points.
- `spar = 1` implies maximum smoothing, so a single cubic curve (i.e. a cubic regression) is fitted to the entire data set.

A drawback of this type of curve-fitting is that it does not provide a mathematical model and therefore we do not have parameters to help with interpretation.

The following code fits a smooth spline to the data and plots milk yield vs days after birth. To see the effect of increasing the amount of smoothing we consider `spar = 0.4` (blue curve) and `spar = 0.8` (green curve).

```
smsp4 <- with(lact, smooth.spline(Day, Yield, spar=0.4))
smsp8 <- with(lact,smooth.spline(Day,Yield, spar=0.8))
fitted4 <- fitted(smsp4)  # extract the fitted values
fitted8 <- fitted(smsp8)
with(lact,plot(Day,Yield))
points(lact$Day, fitted4, type="l", col="blue")
points(lact$Day, fitted8,type="l",col="green")
```



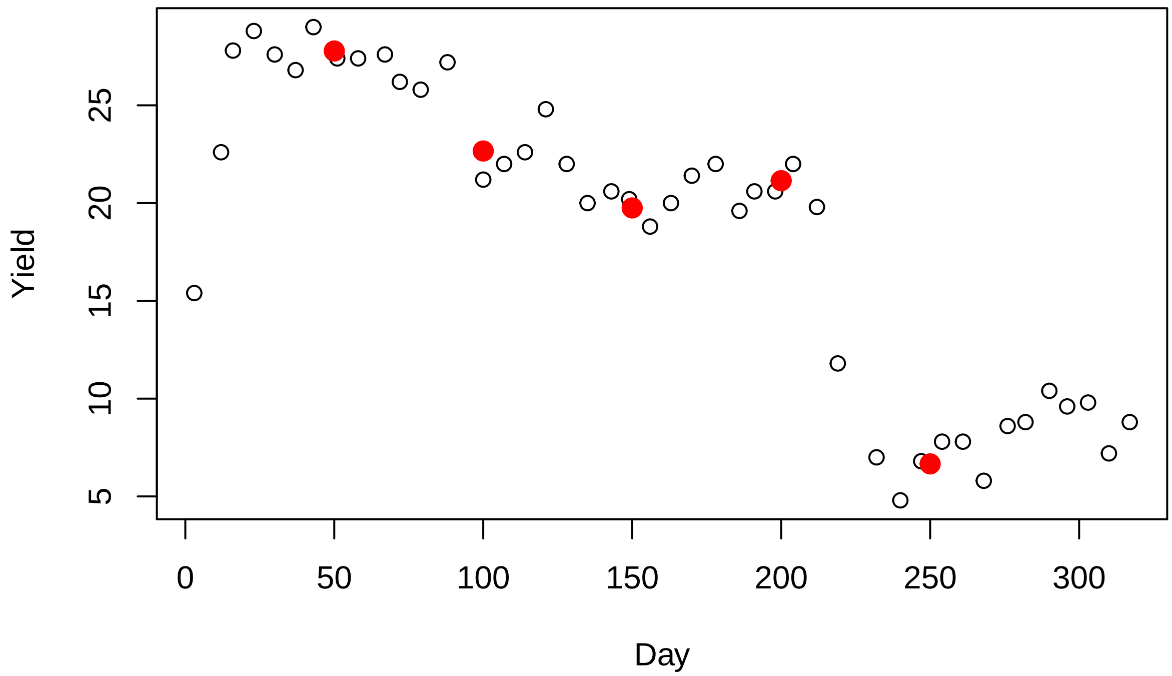
Note that we can predict from a smooth spline without having to put the new data into a `data.frame`, as we did for `nls` and `lm` models. Using the model fitted with `spar = 0.4` we predict milk yields at 5 new days (predicted values plotted in red):

```
newDay <- c(50,100,150,200,250)
pred4 <- predict(smsp4,newDay)
```

```
pred4
```

```
$x
[1]  50 100 150 200 250

$y
[1] 27.7771 22.6639 19.7529 21.1450  6.6635
```

```
with(lact,plot(Day,Yield))
points(pred4$x, pred4$y, pch=20, cex=2, col="red")
```



8.4 Transforming the response variable

Sometimes it is appropriate to transform the response variable Y to achieve a linear model. We have already seen the log transformation:

$$\ln Y = a + bX + \epsilon$$

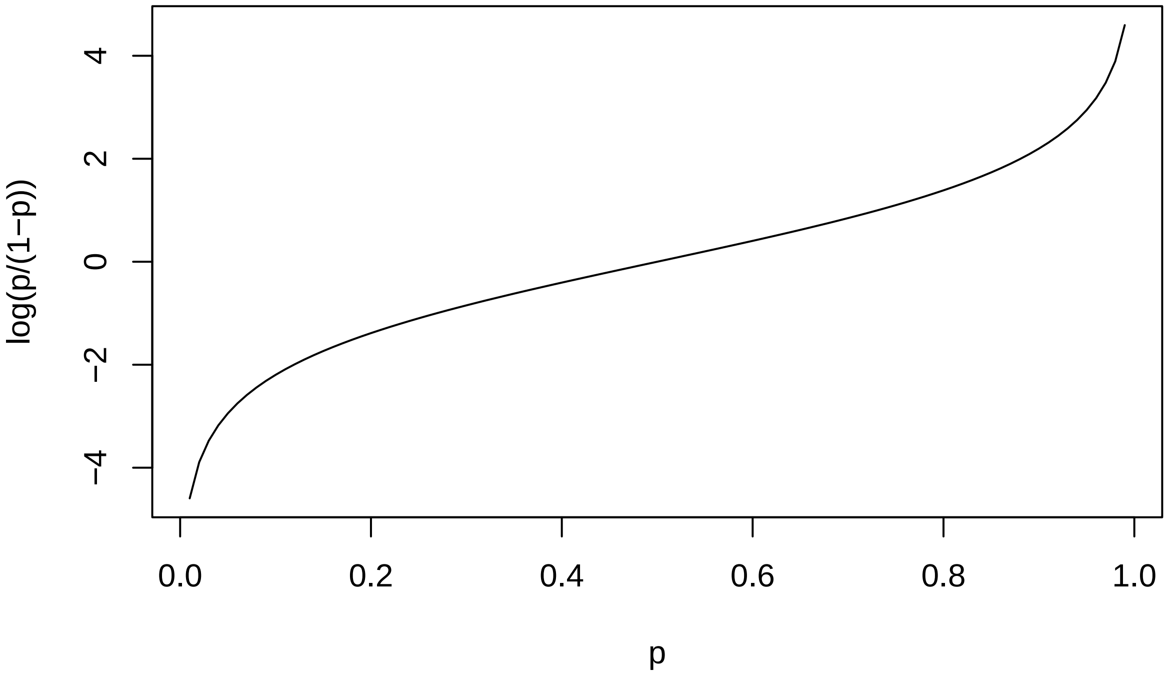
Simple transformations of this type are legitimate provided the resulting residuals satisfy the necessary assumptions regarding normality, independence and equal variance at all values of X. Apart from the log for continuous data or the square root, a well known transformation is the *logit*, used when the data are proportions or percentages, such that the Y values lie in the range $[0, 1]$ or $[0, 100]$. Denoting the proportions by p_i, the definition is

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right)$$

Note that the logit transformation is only defined for values of p_i in $(0, 1)$; that is, values at the extremes are not allowed.

```
# create a sequence of values in [0.01, 0.99] in steps of 0.01
p <- seq(0.01,0.99,0.01)
```

```r
logit <- log(p/(1-p))         # derive the logits of p
plot(p,logit,type="l",ylab="log(p/(1-p))")
```



Logits stretch the response values at the extremes, that is near zero and 1, where interest often lies and small differences really matter. The use of logits for proportion data does not guarantee that the residuals will have the desired properties.

We illustrate with data on the age of menarche for a sample of girls. Each girl was asked the date of her first period and the results grouped by age. Within each age group we have the total number of girls (Total), the number who had their first period (*Menarche*) and the mid-point of the age range for that group (*Age*).

```r
library(MASS)
head(menarche)
```

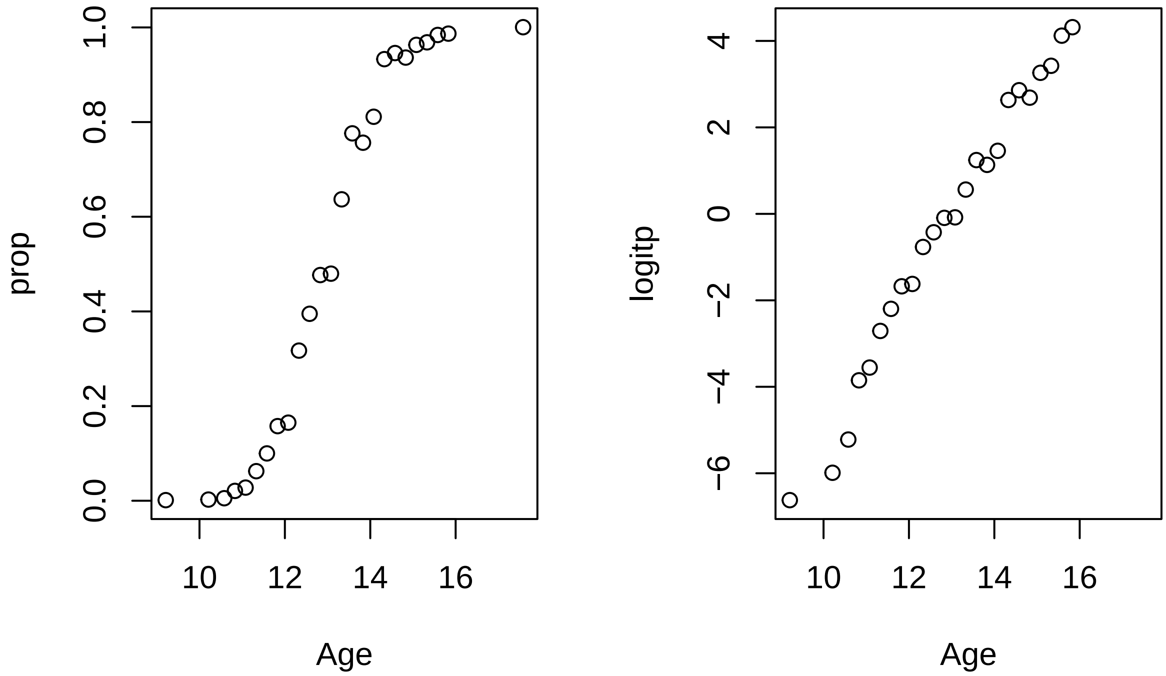
```
    Age Total Menarche      prop  logitp
1  9.21   376        0 0.0013298 -6.6214
2 10.21   200        0 0.0025000 -5.9890
3 10.58    93        0 0.0053763 -5.2204
4 10.83   120        2 0.0208333 -3.8501
5 11.08    90        2 0.0277778 -3.5553
6 11.33    88        5 0.0625000 -2.7081
```

We force the proportion into $(0, 1)$ in this example by simply adding a small positive amount for illustration.

```r
menarche$prop <- with(menarche, (Menarche+0.5)/(Total))
menarche$logitp <- with(menarche, log(prop/(1-prop)))
```

```
Warning in log(prop/(1 - prop)): NaNs produced
```

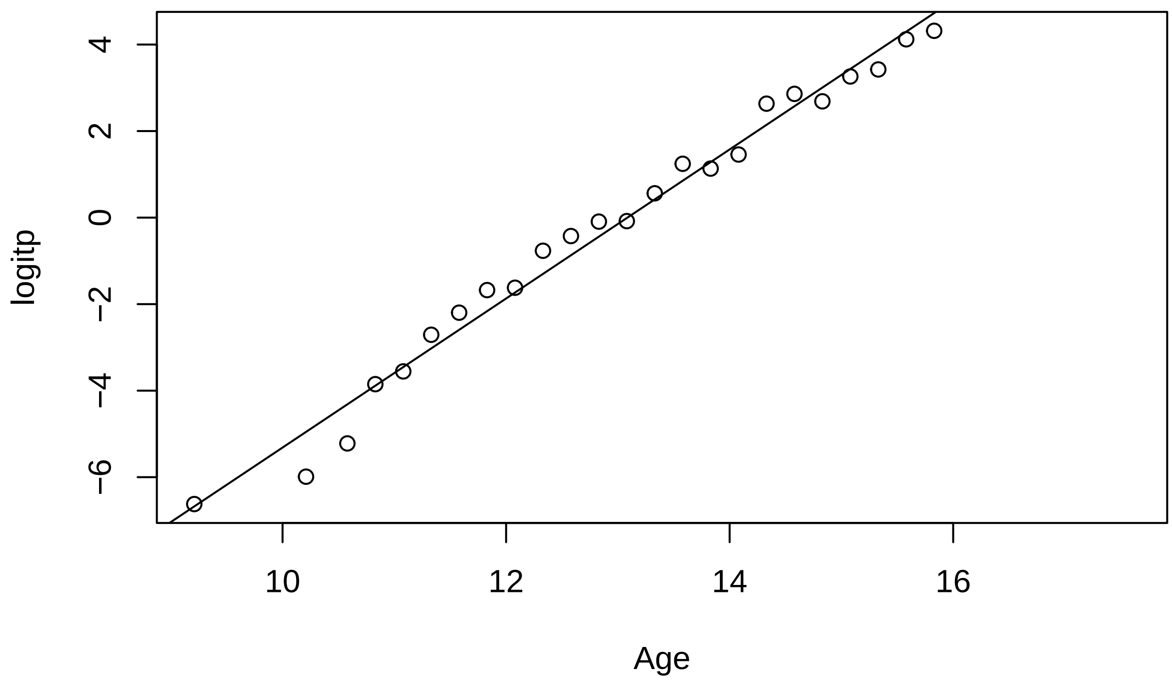
```r
par(mfrow=c(1,2))
with(menarche, plot(Age,prop))
with(menarche, plot(Age,logitp))
```




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

It looks like a linear model might be a sensible option to fit the transformed data.

```
m.men.1 <- lm(logitp~Age,data=menarche)
with(menarche, plot(Age,logitp))
abline(m.men.1)
```



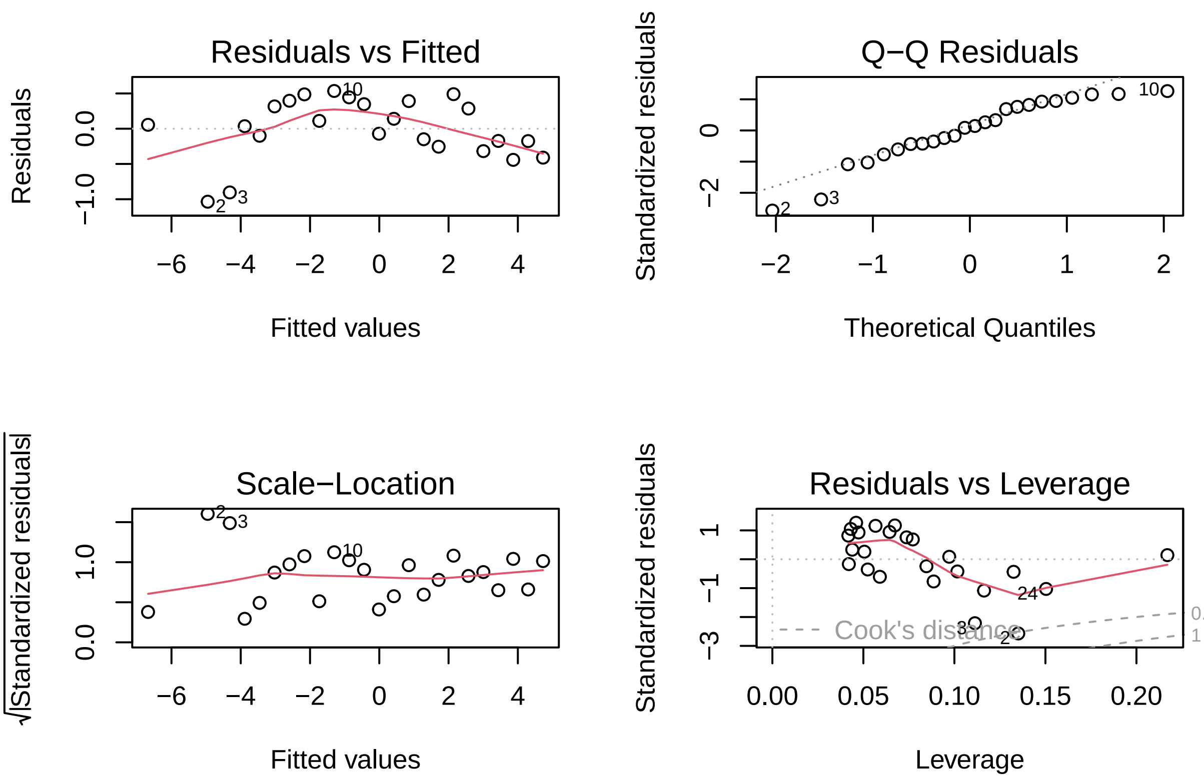

```
summary(m.men.1)
```

```
Call:
```

```
lm(formula = logitp ~ Age, data = menarche)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0353 -0.1955  0.0452  0.3586  0.5349

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -22.543      0.638   -35.3   <2e-16 ***
Age            1.723      0.049    35.2   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.433 on 22 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.983, Adjusted R-squared:  0.982
F-statistic: 1.24e+03 on 1 and 22 DF,  p-value: <2e-16
```

While this analysis looks reasonable at first sight the diagnostic plots (see below) show that there is some room for improvement.

```r
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.men.1)
```




```r
# Revert to original settings
par(oldpar)
```

The q-q plot shows that the residuals do not seem to clearly fit a normal distribution. A popular test of normality is the Shapiro-Wilk's test. The null hypothesis is normality.

```
shapiro.test(residuals(m.men.1))
```

```
    Shapiro-Wilk normality test

data:  residuals(m.men.1)
W = 0.923, p-value = 0.069
```

Both diagnostic tools suggest that the assumption of normality of the residuals might be questionable, however the resulting line might still be good enough for practical purposes.

8.5 Exercises

1. For the `ox.xls` data example, fit a smooth spline and decide whether this is a better curve to use than the Gompertz curve. One approach is to compare the residual versus fitted plots, another is to compare the variances of the two sets of residuals.
2. The file `alloystrength.txt` records the compressive strength of an alloy fastener used in the construction of aircraft. Ten pressure loads, in psi, were used and different numbers of fasteners tested at each load. Is it reasonable to say that the proportion of fasteners failing is directly linearly related to the pressure load? If not, what is a more suitable transformation/model? Derive a 95% confidence interval for the proportion expected to fail at a pressure of 4000psi. At what pressure is there a 50% chance of failure?
3. The file `Dilution.txt` shows the optical density of four solutions at a sequence of dilution values. For any one of the four dilutions plot the optical against dilution, then against log10(dilution). Fit a logistic curve to the optical density values as a response to log10 dilutions and obtain 95% confidence intervals for all 4 parameter values. What are the upper and lower asymptotes? Derive a 95% confidence interval for the mean optical density at the point of inflexion of this curve. Does this curve describe the data satisfactorily and if so how would you quantify how well this logistic model describes the data?

9 Multiple linear regression

9.1 Fitting the model

Instead of simple regression models of the form $Y = a + bX + \epsilon$ we now consider several independent X variables

$$Y = b_0 + b_1 X_1 + ... + b_p X_p + \epsilon$$

Again the residual errors represented by ϵ are assumed to be independent and normally distributed with mean 0 and constant variance σ^2. Alternatively, we can say that $Y \sim N(\mu, \sigma^2)$ where $\mu = b_0 + b_1 X_1 + ... + b_p X_p$.

We illustrate multiple regression using the `savings` data from the R package `faraway`.

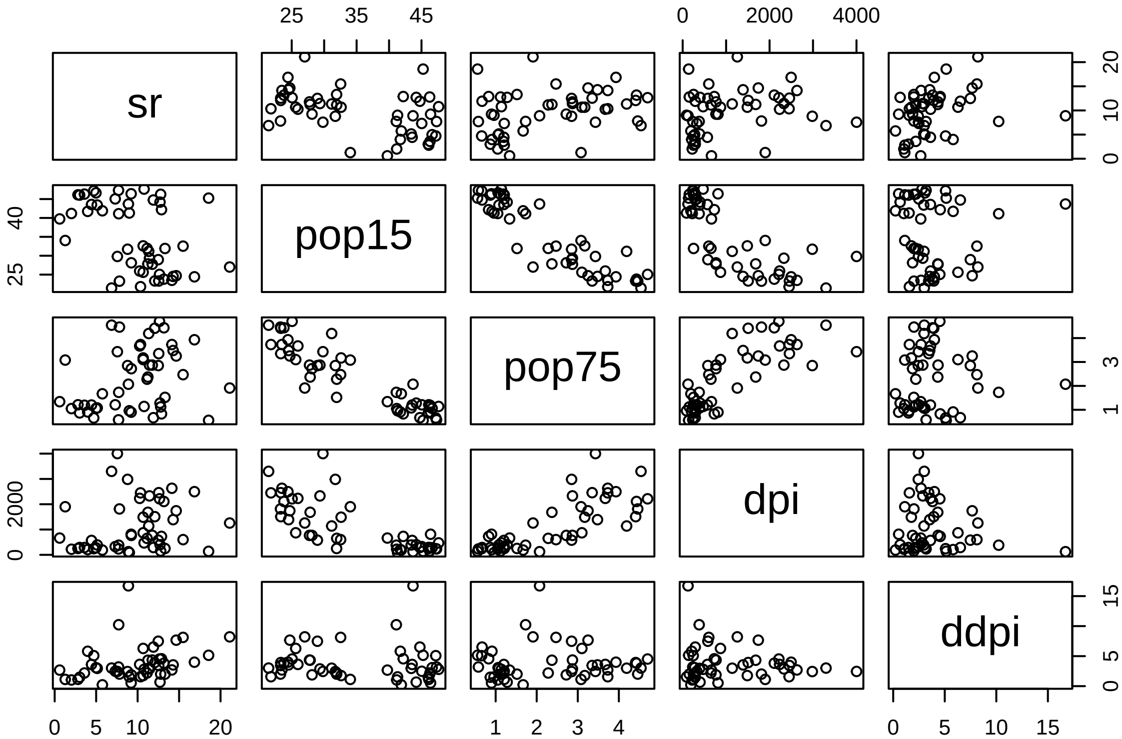
```
# install.packages("faraway")
library(faraway)
str(savings)
```

```
'data.frame':	50 obs. of  5 variables:
 $ sr   : num  11.43 12.07 13.17 5.75 12.88 ...
 $ pop15: num  29.4 23.3 23.8 41.9 42.2 ...
 $ pop75: num  2.87 4.41 4.43 1.67 0.83 2.85 1.34 0.67 1.06 1.14 ...
 $ dpi  : num  2330 1508 2108 189 728 ...
 $ ddpi : num  2.87 3.93 3.82 0.22 4.56 2.43 2.67 6.51 3.08 2.8 ...
```

These data are savings rates, *sr*, for 50 countries, averaged across 1960-70. The variable *dpi* is disposable, per-capita income in US dollars; *ddpi* is the % rate of change in per capita disposable income, *sr* is aggregate personal saving divided by disposable income. The *pop15* and *pop75* columns are the percentages of the population under 15 and over 75.

As always, before carrying out a regression analysis, some plotting and calculation of summary statistics should be done to familiarise yourself with the data and to identify unexpected or odd features. We can do pairwise scatterplots of variables from the savings data set.

```
pairs(savings)
```



The correlation matrix quantifies the extent of the linearity in these pairwise plots.

```
options.default<-options()
options(digits=2)
cor(savings)
```

```
         sr  pop15  pop75   dpi   ddpi
sr     1.00 -0.456  0.317  0.22  0.305
pop15 -0.46  1.000 -0.908 -0.76 -0.048
pop75  0.32 -0.908  1.000  0.79  0.025
dpi    0.22 -0.756  0.787  1.00 -0.129
ddpi   0.30 -0.048  0.025 -0.13  1.000
```

```
options(options.default)
```

Note that *sr* is not highly correlated with any one of the other variables but these other variables do have some high correlations among themselves. Now we can proceed with the regression analysis.

```
m.sr.1 <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
summary(m.sr.1)
```

```

Call:
lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)

Residuals:
   Min     1Q Median     3Q    Max 
-8.242 -2.686 -0.249  2.428  9.751 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 28.566087   7.354516    3.88  0.00033 ***
pop15       -0.461193   0.144642   -3.19  0.00260 ** 
```

```
pop75       -1.691498   1.083599   -1.56  0.12553
dpi         -0.000337   0.000931   -0.36  0.71917
ddpi         0.409695   0.196197    2.09  0.04247 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.8 on 45 degrees of freedom
Multiple R-squared:  0.338, Adjusted R-squared:  0.28
F-statistic: 5.76 on 4 and 45 DF,  p-value: 0.00079
```

The fitted regression is

$$sr = 28.6 - 0.46pop15 - 1.69pop75 - 0.0003dpi + 0.41ddpi$$

The interpretation of the regression coefficients in multiple regression is identical to the interpretation for simple regression. For example *sr* decreases by 1.69 for every increase of size 1 in pop75. Again, the intercept, 28.6, is the value of *sr* when all the independent variables are zero (a situation which cannot occur in practice).

We begin by asking whether it is worth having all four X variables in the model; in other words whether $b_1 = b_2 = b_3 = b_4 = 0$. This is assessed by the F test statistic at the end of the summary output ($F = 5.756$) that has associated a very small p-value $p = 0.0007904$. Thus, the probability of obtaining an F ratio (signal to noise) as extreme as 5.7 when all the regression coefficients are zero is $p = 0.0008$. This is so unlikely that we conclude that at least one of the coefficients is not zero.

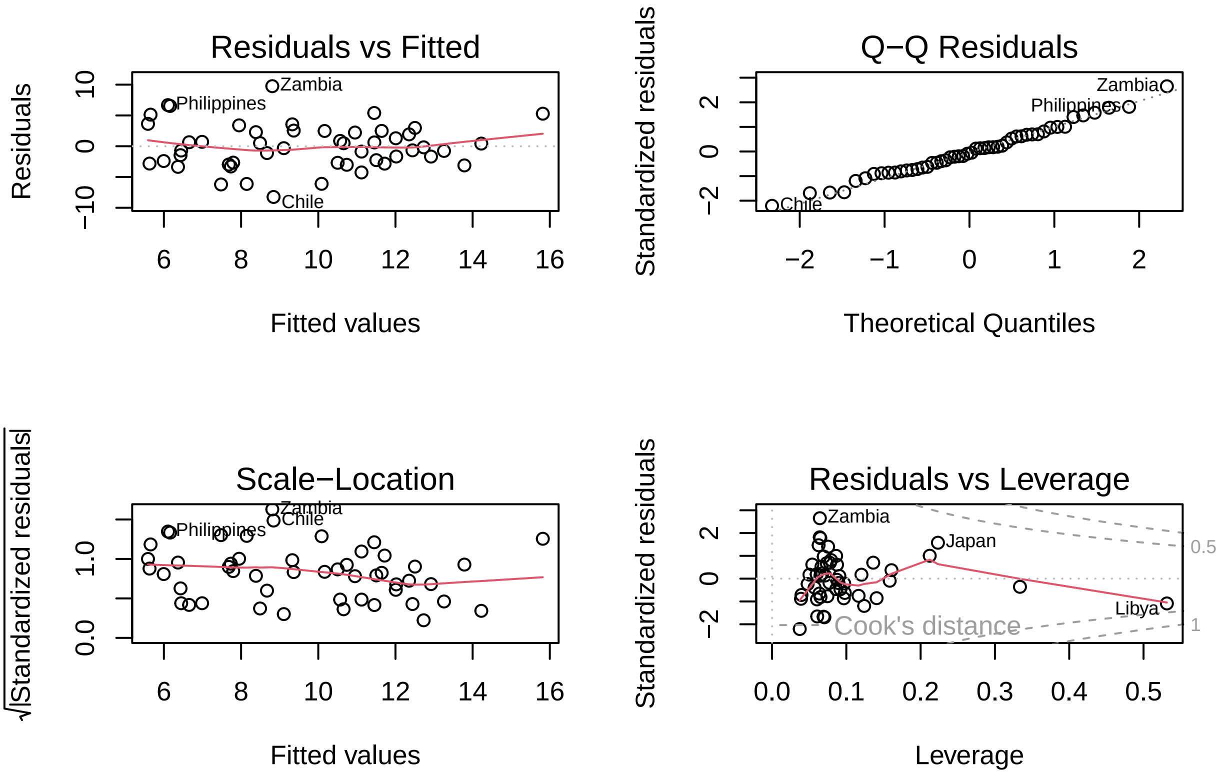
The model assumes that the residual term ϵ has a normal distribution with mean zero and an (unknown) variance σ^2. The *residual standard error* in the results, 3.803, is the estimate of σ.

The significance levels for each term in the regression, Pr(>|t|), test the null hypothesis that this term is zero (i.e. the coefficient $b_i = 0$), given that all the other terms are already in the model. It is important to distinguish between testing whether X_i affects Y (which we would do by simple linear regression), and whether X_i affects Y after all the other X variables have been included in the model. The results above test the latter hypothesis. And, as said, we can interpret the regression coefficient b_i as the amount Y will increase when X_i increases by 1.

Multiple R-squared and *Adjusted R-squared* are measures of goodness of fit. The closer to 1 these are, the better the fit. There tends to be a larger discrepancy between these two measures in the context of multiple regression than simple regression. Adjusted R-squared is the preferred measure because it takes into account the number of X variables that have been fitted. With the unadjusted multiple R-squared, the more variables you include the more R-squared approaches 1.

All other results from the `lm` function are interpreted in the same way as for simple linear regression and we can generate the same plots and access the same information as before.

```r
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.sr.1)
```




```r
# Revert to original settings
par(oldpar)
```

We can equally obtain confidence intervals for the parameter estimates

```r
confint(m.sr.1,level=.99)
```

```
                  0.5 %     99.5 %
(Intercept)  8.7854902 48.3466829
pop15       -0.8502207 -0.0721656
pop75       -4.6059291  1.2229338
dpi         -0.0028412  0.0021674
ddpi        -0.1179939  0.9373838
```

To predict savings rates for new combinations of *pop15*, *pop75*, *dpi* and *ddpi* we must put these new levels into a `data.frame` object.

```r
newvalues <- data.frame(pop15=c(32,33), pop75=c(3,2), dpi <- c(700,700),ddpi=c(3,3))
ci <- predict(m.sr.1,newvalues,interval="confidence")
pi <- predict(m.sr.1,newvalues,interval="prediction")
```

The confidence interval tells us about the average value of *sr*. The prediction interval tells us about a single value of *sr*. Notice how the prediction interval is much wider than the confidence interval.

9.2 Testing subsets of the X variables

We can test a wide range of hypotheses on the explanatory variables such as:

- Is it worth including any X variables in the model?
- Could X_3 reasonably be dropped from the model?
- Could X_2 and X_3 be dropped simultaneously from the model without loss?
- Do X_1 and X_4 have the same coefficients?

- Is it reasonable to force the coefficient of one of the X's to be 1?

The first of these is the F-test we have already met. All of these example hypotheses have one feature in common: they can all be expressed as a null hypothesis which is a subset of an alternative Hypothesis *full model*.

Thus, for the second example,

$$H_0 : mean(Y) = b_0 + b_1 X_1 + b_2 X_2 + \quad + b_4 X_4$$
$$H_A : mean(Y) = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4$$

Or H_0 could also be written as

$$H_0 : b_3 = 0$$

For the third example,

$$H_0 : mean(Y) = b_0 + b_1 X_1 + \quad + b_4 X_4$$
$$H_A : mean(Y) = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4$$

Or

$$H_0 : b_2 = b_3 = 0$$

The test statistic for hypothesis comparisons of this type is based on the residual sums of squares (RSS). For any regression we have *actual* observations Y, *fitted* (estimated) observations $\hat{Y}$ and the *residuals* $(Y - \hat{Y})$. The RSS $\sum(Y - \hat{Y})^2$ is a measure of how well the fitted model matches the data, low values indicate a good fit.

When H_0 is true the RSS of the submodel (H_0) and the RSS of the full Model (H_A) should be approximately the same. However when H_0 is false we would expect the RSS of the submodel to be noticeably greater than the RSS of the full model. So large values of $RSS(sub) - RSS(full)$ will suggest that the H_0 is false. The test statistic is

$$F = \frac{(RSS(sub) - RSS(full))/(p - q)}{RSS(full)/(n - 1 - p)},$$

where n is the number of observations, p is the number of terms in the full model and q is the number of terms in the submodel.

When H_0 is true this test statistic has an F distribution with $(p-q)$ and $(n-1-p)$ degrees of freedom. Large values of the test statistic are evidence away from the null hypothesis towards the alternative hypothesis.

Returning to the savings data set, we will test whether it is worth having *dpi* in the model. That is, we are going to test

$$H_0 : b_3 = 0$$
$$H_A : b_3 \neq 0$$

```
Ho <- lm(sr~ pop15+pop75+ddpi, data=savings)
Ha <- lm(sr~pop15+pop75+dpi+ddpi, data=savings)
anova(Ho,Ha)
```

```
Analysis of Variance Table

Model 1: sr ~ pop15 + pop75 + ddpi
Model 2: sr ~ pop15 + pop75 + dpi + ddpi
  Res.Df RSS Df Sum of Sq    F Pr(>F)
```

```
1      46 653
2      45 651  1       1.89 0.13   0.72
```

The test statistic in this case is

$$F = \frac{(652.61 - 650.71)/(4-3)}{650.71/(50-1-4)}$$

and

```r
pf(.13, df1=1, df2=45, lower.tail=F)
```

```
[1] 0.72012
```

The result is not statistically significant at the usual 5% significance level. The data are not suggesting to reject the null hypothesis and hence we conclude that *dpi* has no additional effect on savings when the effects of *pop15*, *pop75* and *ddpi* are already taken into account.

Note that we had already seen this result.

```r
summary(m.sr.1)
```

```

Call:
lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)

Residuals:
   Min     1Q Median     3Q    Max 
-8.242 -2.686 -0.249  2.428  9.751 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 28.566087   7.354516    3.88  0.00033 ***
pop15       -0.461193   0.144642   -3.19  0.00260 ** 
pop75       -1.691498   1.083599   -1.56  0.12553    
dpi         -0.000337   0.000931   -0.36  0.71917    
ddpi         0.409695   0.196197    2.09  0.04247 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.8 on 45 degrees of freedom
Multiple R-squared:  0.338, Adjusted R-squared:  0.28 
F-statistic: 5.76 on 4 and 45 DF,  p-value: 0.00079
```

The significance of the additional effect of *dpi* using a t-test is based on a p-value equal to 0.7192, nearly numerically identical to that from the approach above using a F-test. In this particular circumstance, and only in this circumstance, when the *numerator* degrees of freedom is 1, the F statistic is the same as the squared t statistic and generates the same p-value.

9.3 ANOVA summary of multiple regression

Often the results of a regression analysis are presented in an ANOVA table format, as follows.

```r
anova(m.sr.1)
```

```
Analysis of Variance Table

Response: sr
          Df Sum Sq Mean Sq F value  Pr(>F)
```

```
pop15      1    204   204.1    14.12 0.00049 ***
pop75      1     53    53.3     3.69 0.06113 .
dpi        1     12    12.4     0.86 0.35936
ddpi       1     63    63.1     4.36 0.04247 *
Residuals 45    651    14.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first line of the anova table shows the results of testing

$$H_0 : mean(Y) = b_0$$
$$H_A : mean(Y) = b_0 + b_1X_1$$

and shows that X_1 (*pop15*) explains significantly more of the savings ratio than using only the mean of the savings ratio (b_0).

The second line of the table shows the results of testing

$$H_0 : mean(Y) = b_0 + b_1X_1$$
$$H_A : mean(Y) = b_0 + b_1X_1 + b_2X_2$$

and shows that X_2 (*Pop75*) just fails, at 5% significance level, to add significant improvement to the prediction of the savings ratio when X_1 is already in the model.

The third line of the table shows the results of testing

$$H_0 : mean(Y) = b_0 + b_1X_1 + b_2X_2$$
$$H_A : mean(Y) = b_0 + b_1X_1 + b_2X_2 + b_3X_3$$

and shows that X_3 (*dpi*) adds nothing to the prediction of the savings ratio when X_1 and X_2 are already in the model.

Finally the fourth line of the table shows the results of testing

$$H_0 : mean(Y) = b_0 + b_1X_1 + b_2X_2 + b_3X_3$$
$$H_A : mean(Y) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$$

and shows that X_4 (*ddpi*) makes a significant improvement to the prediction of the savings ratio when X_1, X_2 and X_3 are already in the model.

Thus the `anova` shows the *cumulative* effect of adding variables to the regression model. Variables are added in the order specified in the `lm` function.

It is important to be aware of the difference between this summary of the `lm` fit provided by the `anova` function and what is provided by the function `summary` applied on the `lm` object directly. The latter shows the results of testing each variable after all the others have been included in the model, i.e *additional* and not cumulative effects.

9.4 Variable/model selection strategies

Sometimes, when there are several variables for potential inclusion in the model, we wish to adopt a formal procedure for selecting the best subset.

There are varied proposals to do this and it is an active area of research. It is important to check the details of the options implemented by the software package used.

Examples of common procedures are:

- Backward elimination
- Forward selection
- Stepwise regression
- All subsets

With backward elimination we start with all the predictors in the model, then repeatedly remove the predictor with the highest p-value greater than some appropriate, pre-defined cut-off.

Forward selection reverses this approach and starts with no terms in the model. The single term with the smallest p-value beyond the cut-off is added first. Among the remaining variables the one with the strongest p-value beyond the cut-off, when added to the model, is selected. And so on. For this type of procedure the cut-off is usually less stringent than 5% and can be as high as 20%.

Stepwise regression is a combination of the first two approaches and allows the possibility at each step of adding or dropping a variable, whichever has the strongest effect on the F-ratio. Best practice suggests starting both with no terms and with the full model to verify that both starting points lead to the same final model.

Criteria other than the F-ratio significance level can be used to accept or reject variables. Common alternatives are the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Both AIC and BIC criteria are based on various assumptions and asymptotic approximations. They are typically used together in model selection. The lower the value in these measures the better.

The AIC has a built in penalty for models with larger number of parameters. The AICc variant is recommended when sample size is small with respect to the number of model parameters. For nested models, estimates of relative importance of predictor variables can be made by summing the *Akaike weights* (evidence in favor of model i being the actual best model amongst a number of candidate models) of variables across all the models where the variables occur. The main difference in practice between AIC and BIC is the size of the penalty, BIC penalises model complexity more heavily. Thus, AIC always has a chance of choosing too big a model, regardless of the number of observations. BIC has very little chance of choosing too big a model if the number of observations is sufficient, but it has a larger chance than AIC, for any given number of observations, of choosing too small a model.

Another way to compare model specifications statistically is based on the likelihood ratio test (LRT; but limited to comparisons between two models at a time). The LR is a measure of the strength of evidence favouring a model (hypothesis) A over a model (hypothesis) B. In R, functions like `anova` and `drop1` show results from AIC, BIC and LRT together.

Some software allows to search among all possible models to see which subset of variables fits the data best. This approach might take a lot of computing time and it is susceptible to presenting a *best* model which is not amenable to scientific interpretation. A modification of the all subsets method is often used when models are to be used for predictive (rather than explanatory) purposes. The fit of the model is assessed according to some predefined criterion (often the AIC or BIC). It requires that subsets are restricted to those which make scientific sense. Often several models have information criteria similar to the *best* model and the average of all the models within a certain distance of the best information criterion is used (see literature about multimodel inference and model averaging; Kenneth P. Burnham, David R. Anderson (2002). Model selection and multimodel inference: A practical information-theoretic approach. New York: Springer). The R package `MuMIn` implements methods for variable and multi-model selection within this framework.

No matter which model is selected as the *best* by any of the above strategies, its residuals should always be examined by the usual diagnostic plots to ensure the usual regression assumptions hold.

We illustrate an approach to backward elimination using the *state* data set. This data set comes with R and help(state) provides a full description. It is a collection of statistics from 50 US states: per capita income, illiteracy, life expectancy, murder and non-negligent manslaughter rate per 100000 population, percent high school graduates, mean number of days with min temperature less than 32F, and land area. The question of interest is whether we can predict life expectancy from the other variables.

```r
data(state) #load a data set
statedata <- data.frame(state.x77, row.names=state.abb, check.names=TRUE)
# Model Life.Exp using ALL the variables in the data set
m.be1 <- lm(Life.Exp ~ . , data=statedata)
summary(m.be1)
```

```

Call:
lm(formula = Life.Exp ~ ., data = statedata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.4890 -0.5123 -0.0275  0.5700  1.4945 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  7.09e+01   1.75e+00   40.59  < 2e-16 ***
Population   5.18e-05   2.92e-05    1.77    0.083 .  
Income      -2.18e-05   2.44e-04   -0.09    0.929    
Illiteracy   3.38e-02   3.66e-01    0.09    0.927    
Murder      -3.01e-01   4.66e-02   -6.46  8.7e-08 ***
HS.Grad      4.89e-02   2.33e-02    2.10    0.042 *  
Frost       -5.74e-03   3.14e-03   -1.82    0.075 .  
Area        -7.38e-08   1.67e-06   -0.04    0.965    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.745 on 42 degrees of freedom
Multiple R-squared:  0.736, Adjusted R-squared:  0.692 
F-statistic: 16.7 on 7 and 42 DF,  p-value: 2.53e-10
```

Area has the largest *p*-value so we begin by eliminating it.

```r
m.be2 <- update(m.be1, . ~ . -Area)
summary(m.be2)
```

```

Call:
lm(formula = Life.Exp ~ Population + Income + Illiteracy + Murder + 
    HS.Grad + Frost, data = statedata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.4905 -0.5253 -0.0255  0.5716  1.5037 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  7.10e+01   1.39e+00   51.17  < 2e-16 ***
Population   5.19e-05   2.88e-05    1.80    0.079 .  
Income      -2.44e-05   2.34e-04   -0.10    0.917    
Illiteracy   2.85e-02   3.42e-01    0.08    0.934    
Murder      -3.02e-01   4.33e-02   -6.96  1.5e-08 ***
HS.Grad      4.85e-02   2.07e-02    2.35    0.024 *  
Frost       -5.78e-03   2.97e-03   -1.94    0.058 .  
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.736 on 43 degrees of freedom
Multiple R-squared:  0.736, Adjusted R-squared:  0.699
F-statistic:   20 on 6 and 43 DF,  p-value: 5.36e-11
```

The adjusted R^2 has hardly changed so eliminating *Area* has not given us a poorer fit. We next remove *Illiteracy*.

```r
m.be3 <- update(m.be2,.~.-Illiteracy)
summary(m.be3)
```

```
Call:
lm(formula = Life.Exp ~ Population + Income + Murder + HS.Grad +
    Frost, data = statedata)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4892 -0.5122 -0.0329  0.5645  1.5166

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.11e+01   1.03e+00   69.07  < 2e-16 ***
Population   5.11e-05   2.71e-05    1.89    0.066 .
Income      -2.48e-05   2.32e-04   -0.11    0.915
Murder      -3.00e-01   3.70e-02   -8.10  2.9e-10 ***
HS.Grad      4.78e-02   1.86e-02    2.57    0.014 *
Frost       -5.91e-03   2.47e-03   -2.39    0.021 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.728 on 44 degrees of freedom
Multiple R-squared:  0.736, Adjusted R-squared:  0.706
F-statistic: 24.5 on 5 and 44 DF,  p-value: 1.02e-11
```

The adjusted R^2 has now improved slightly despite dropping *Illiteracy*. *Income* is next to go.

```r
m.be4 <- update(m.be3,.~.-Income)
summary(m.be4)
```

```
Call:
lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
    data = statedata)

Residuals:
   Min     1Q Median     3Q    Max
-1.471 -0.535 -0.037  0.576  1.507

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.10e+01   9.53e-01   74.54  < 2e-16 ***
Population   5.01e-05   2.51e-05    2.00    0.052 .
Murder      -3.00e-01   3.66e-02   -8.20  1.8e-10 ***
HS.Grad      4.66e-02   1.48e-02    3.14    0.003 **
Frost       -5.94e-03   2.42e-03   -2.46    0.018 *
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.72 on 45 degrees of freedom
Multiple R-squared:  0.736, Adjusted R-squared:  0.713
F-statistic: 31.4 on 4 and 45 DF,  p-value: 1.7e-12
```

Note how the adjusted R^2 has improved again. It is not obvious whether *Population* should be removed.

We have used a selection criterion based on significance levels. There are other criteria we might have used, for example improvement in adjusted R^2. As said above, a popular criterion is AIC.

$$AIC = n\ln(RSS/n) + 2p$$

We want to minimise the AIC. The AIC assesses a model by its RSS, accounting for the number of observations n and adding a penalty equal to twice the number of fitted parameters, p. The AIC is good for comparing regression models. However, as a measure in its own right it does not tell us anything directly about the current model being fitted, i.e. how good it is in itself, the criteria just allows to compare its relative merit compared to others, but they might be all poor models. The adjusted R^2 on the other hand, does tell us about the current model being fitted, the closer to 1 the better the fit.

The function `step` performs forward selection, backward elimination or stepwise regression automatically, using AIC rather than p-values. We illustrate backwards elimination below.

```
step(m.be1,direction="backward")
```

```
Start:  AIC=-22.18
Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
    Frost + Area

             Df Sum of Sq  RSS   AIC
- Area        1      0.00 23.3 -24.2
- Income      1      0.00 23.3 -24.2
- Illiteracy  1      0.00 23.3 -24.2
<none>                    23.3 -22.2
- Population  1      1.75 25.0 -20.6
- Frost       1      1.85 25.1 -20.4
- HS.Grad     1      2.44 25.7 -19.2
- Murder      1     23.14 46.4  10.3

Step:  AIC=-24.18
Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
    Frost

             Df Sum of Sq  RSS   AIC
- Illiteracy  1      0.00 23.3 -26.2
- Income      1      0.01 23.3 -26.2
<none>                    23.3 -24.2
- Population  1      1.76 25.1 -22.5
- Frost       1      2.05 25.3 -22.0
- HS.Grad     1      2.98 26.3 -20.2
- Murder      1     26.27 49.6  11.6

Step:  AIC=-26.17
Life.Exp ~ Population + Income + Murder + HS.Grad + Frost

             Df Sum of Sq  RSS   AIC
```

```
- Income      1       0.0 23.3 -28.2
<none>                    23.3 -26.2
- Population  1       1.9 25.2 -24.3
- Frost       1       3.0 26.3 -22.1
- HS.Grad     1       3.5 26.8 -21.2
- Murder      1      34.7 58.0  17.5

Step:  AIC=-28.16
Life.Exp ~ Population + Murder + HS.Grad + Frost

             Df Sum of Sq  RSS   AIC
<none>                    23.3 -28.2
- Population  1       2.1 25.4 -25.9
- Frost       1       3.1 26.4 -23.9
- HS.Grad     1       5.1 28.4 -20.2
- Murder      1      34.8 58.1  15.5


Call:
lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
    data = statedata)

Coefficients:
(Intercept)   Population       Murder      HS.Grad        Frost
   7.10e+01     5.01e-05    -3.00e-01     4.66e-02    -5.94e-03
```

At the start stage we see that the full model has $AIC = -22.185$. The model with *Area* removed has $AIC = -24.182$, an improvement. The model with *Population* removed has $AIC = -20.560$, which is not as good as the full model. And so on. Removing *Area* generates the best value for AIC and leads to the subsequent step. This continues while there is an improvement. In this case the sequence of variable removals is very similar to our previous analysis using p-values rather than AIC.

9.5 Exercises

1. Using the savings data from Faraway's book, multiply the variable *dpi* by 10. What changes does this produce in the regression results? Do these changes make sense?
2. The *hills* data set from the R library MASS shows the record times taken to race to the top of 35 hills, along with the race distance and total height climbed. Plot the data and derive the correlations among these variables. Derive the linear regression of time as a function of distance and height. Is this a good model? Using this model, how much extra time is added to the record for every 100 additional feet in height?
3. The gala data from Faraway records the number of species of tortoise on the various Galapagos islands. There are 30 islands. Endemics is the number of endemic species, Elevation is the highest point in metres on the island, Nearest is the distance (km) from the nearest island, Scrux is the distance (m) from Santa Cruz and Adjacent is the area of the adjacent island. Fit a regression model to predict the number of species on an island as a function of all the other variables. Do we need all the other variables? Is your model a good fit? Derive confidence and prediction intervals for the number of species when Endemics and all the other independent variables have values equal to their mean. What is the difference in interpretation for these two intervals?
4. Using the savings data from Faraway's book investigate whether it is worth having *ddpi* in the model when *dpi*, *Pop15* and *Pop75* are already included. Use the same hypothesis testing procedure as the notes used for *dpi* and confirm your result against the `lm` output for the model which includes all four X variables.

5. Use backward elimination and p-values (i.e. use `update`) to determine how many variables should be included in the *gala* regression model. Do you get the same results using the AIC criterion (i.e. use `step`)? Use `step( . . , direction= "both")` to perform a stepwise selection for these data. What is a good final model with this approach?

10 Generalised linear models

10.1 Introduction

So far we have assumed that our response data are at least approximately normally distributed, though we sometimes have had to transform to achieve this. Another common feature of our approach so far has been to seek a linear relationship between the response Y and the X variables, again perhaps after a transformation.

The theory of generalised linear models (GLMs) enables us to expand the range of distributions for the dependent response (Y) and to incorporate, as standard, a wide range of transformations linking the Y variable to the X variables. Although the underlying theory is now more complex, it provides a single unifying algorithm by which all the coefficients in our regression models can be estimated and all inferences can be made. GLMs can be thought of as introducing more flexibility to the multiple regression model we have used so far.

The first distinctive feature of GLMs is the distribution of the response variable Y. GLMs allow for a normal distribution (as in ordinary multiple regression), but also for other distributions like the binomial and Poisson distributions for binary and count response variables respectively. Each of these is referred to as distribution *family.*

The second feature is a function (or transformation) of the mean value of Y which links it to a linear combination of the X variables. Let μ denote the mean of Y and g denote the *link function.* We then have

$$g(\mu) = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_p X_p$$

This is similar, but not identical, to regression models we have already seen after a transformation of the Y variable. The following default pairings of distribution family and link function are both theoretically convenient and practically useful.

Data distribution	**Type of data**	**Link function**
Normal	Continuous	identity (i.e. none)
Poisson	Counts	log(μ)
Binomial	'r successes from n trials'	Logit, log(p/(1-p))

The third distinguishing feature, which is a consequence of the first two extensions to the basic linear regression model, is that the variance of Y becomes a known function of the mean of Y, μ. Denoting this function of μ by V, we have

$$Variance(Y) = \sigma^2 V(\mu),$$

where σ^2 is called the dispersion parameter. It is a constant or scaling factor. For the Poisson and binomial distributions it is 1. For the normal it must be estimated from the data. In practice, for GLMs, we must specify the distribution family and the link function. After that the software package generates the same type of estimates, p-values and diagnostic plots that ordinary regression models generate. However, the underlying methodology is different and the statistical tests are different.

10.2 Deviance

With GLMs the deviance has the same pivotal role in hypothesis testing as residual sums of squares (RSS) have in ordinary regression. The deviance measures the discrepancy between the current model and the saturated model, i.e. the model which fits the data perfectly (the RSS does exactly the same). The deviance is based on so-called likelihood functions, whereas the RSS is based on fitted values. The smaller the deviance the better the current model fits the data. For GLMs with normal error and identity link the deviance is identical to the RSS. All other GLMs have their own formula for the deviance. One consequence of these formulae for the binomial and Poisson families is that the *mean deviance*, i.e. *deviance/df(deviance)*, should be approximately equal to 1.

10.3 Simple linear regression as a GLM

We return to the *Diffpor* data used to illustrate simple linear regression. Now we analyse these data as a GLM. Note that Gaussian is a synonym for normal and the default link function is *identity*.

```
diffpor <- read.table("diffpor.txt",header=TRUE)
m.reg.1 <- lm(Diffusivity~Porosity, data=diffpor)
m.glm.1 <- glm(Diffusivity~Porosity, family=gaussian(link=identity), data=diffpor)
```

Note that `summary(m.glm.1)` and `plot(m.glm.1)` generate almost identical output to `summary(m.reg.1)` and `plot(m.reg.1)` used before. However, they present information about σ^2 differently (look for "Dispersion parameter for Gaussian family …" in the first and "Residual standard error …" in the second).

```
summary(m.glm.1)
```

```

Call:
glm(formula = Diffusivity ~ Porosity, family = gaussian(link = identity),
    data = diffpor)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -36.15       7.06   -5.12  5.3e-06 ***
Porosity      191.87      19.02   10.09  1.9e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 72.209)

    Null deviance: 10811  on 49  degrees of freedom
Residual deviance:  3466  on 48  degrees of freedom
AIC: 359.8

Number of Fisher Scoring iterations: 2
```

```
summary(m.reg.1)
```

```

Call:
lm(formula = Diffusivity ~ Porosity, data = diffpor)

Residuals:
    Min      1Q  Median      3Q     Max
-22.838  -3.652  -0.259   6.451  15.083

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -36.15       7.06   -5.12  5.3e-06 ***
Porosity      191.87      19.02   10.09  1.9e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 8.5 on 48 degrees of freedom
Multiple R-squared:  0.679, Adjusted R-squared:  0.673
F-statistic:  102 on 1 and 48 DF,  p-value: 1.92e-13
```

The function `glm` provides estimates of the regression coefficients and the dispersion parameter. Although `summary` provides inferences based on t-tests derived from asymptotic theory, tests based on the model deviances are generally preferable for GLMs. Conceptually these work in exactly the same way as the tests based on RSS and are available from the `anova` function.

```r
anova(m.glm.1, test='F')
```

```
Analysis of Deviance Table

Model: gaussian, link: identity

Response: Diffusivity

Terms added sequentially (first to last)

         Df Deviance Resid. Df Resid. Dev   F  Pr(>F)
NULL                        49      10811
Porosity  1     7345        48       3466 102 1.9e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(m.reg.1)
```

```
Analysis of Variance Table

Response: Diffusivity
          Df Sum Sq Mean Sq F value  Pr(>F)
Porosity   1   7345    7345     102 1.9e-13 ***
Residuals 48   3466      72
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

'NULL' here refers to a model based only on an overall mean: $Mean(Diff) = b_0$, its residual deviance is 10811.

'Porosity' here refers to the model: $Mean(Diff) = b_0 + b_1 Porosity$ and its residual deviance is 3466.

'NULL' is a sub-model of the 'Porosity' model. We can think of the 'Porosity' model as H_A and the 'NULL' model as H_0 in hypothesis testing. In a previous section we said "… large values of $RSS(sub) - RSS(full)$ suggest H_A is true. Small values suggest H_0 is true."

For GLMs we say: "large values of $Deviance(sub) - Deviance(full)$ suggest H_A is true. Small values suggest H_0 is true". This difference in deviances is referred to in this example as the *deviance due to Porosity*.

Only because we are using the Gaussian family with an identity link function, i.e the same model we used previous ($Y = a + bX + \epsilon$), the residual deviance results here are identical to the residual sums of squares from the ordinary regression analysis.

Note that it is not always correct to say that the smaller the residual deviance the better the model describes the data. Assumptions regarding distribution and variances still have to be assessed.

10.4 Binomial GLM (logistic) regression

Returning to the menarche data set we used before, for each age group we have the total number of girls and the number among them who have had their first period.

```
library(MASS)
str(menarche)
```

```
'data.frame':   25 obs. of  5 variables:
 $ Age     : num  9.21 10.21 10.58 10.83 11.08 ...
 $ Total   : num  376 200 93 120 90 88 105 111 100 93 ...
 $ Menarche: num  0 0 0 2 2 5 10 17 16 29 ...
 $ prop    : num  0.00133 0.0025 0.00538 0.02083 0.02778 ...
 $ logitp  : num  -6.62 -5.99 -5.22 -3.85 -3.56 ...
```

We want to model p, the probability a girl having her first period, as a function of a single X variable, Age. The model would be

$$Y \sim Bin(N, p)$$
$$\text{logit}(p) = b_0 + b_1 Age$$

The response is *Menarche* which has a binomial distribution with parameters N and p. The N, the column *Total*, varies with each value of *Age*. It is worth noting that although the response is *Menarche*, we are modelling p, the expected (mean) value of Y as linear function of the explanatory variables.

We need to create appropriate structures for the function `glm` when the data are binomial.

```
yes <- menarche$Menarche
no <- with(menarche,Total-Menarche)
y <- cbind(yes, no) #create a single 2-column object as the response
m.men2.glm <- glm(y ~ Age, family=binomial(link="logit"), data=menarche)
summary(m.men2.glm)
```

```

Call:
glm(formula = y ~ Age, family = binomial(link = "logit"), data = menarche)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -21.226      0.771   -27.5   <2e-16 ***
Age            1.632      0.059    27.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3693.884  on 24  degrees of freedom
Residual deviance:   26.703  on 23  degrees of freedom
AIC: 114.8

Number of Fisher Scoring iterations: 4
```

```
confint(m.men2.glm)
```

```
Waiting for profiling to be done...
               2.5 %   97.5 %
(Intercept) -22.7866 -19.7638
```

```
Age             1.5201   1.7513
```

The coefficient for *Age* is clearly significant. However for GLMs this t-test, derived from asymptotic theory as said before, is sometimes unreliable. A better statistical test, based on changes in the deviance, is given in the `anova` summary. However, for binomial and Poisson data, where the dispersion parameter must be 1, we have to use Chi-squared (χ^2) tests.

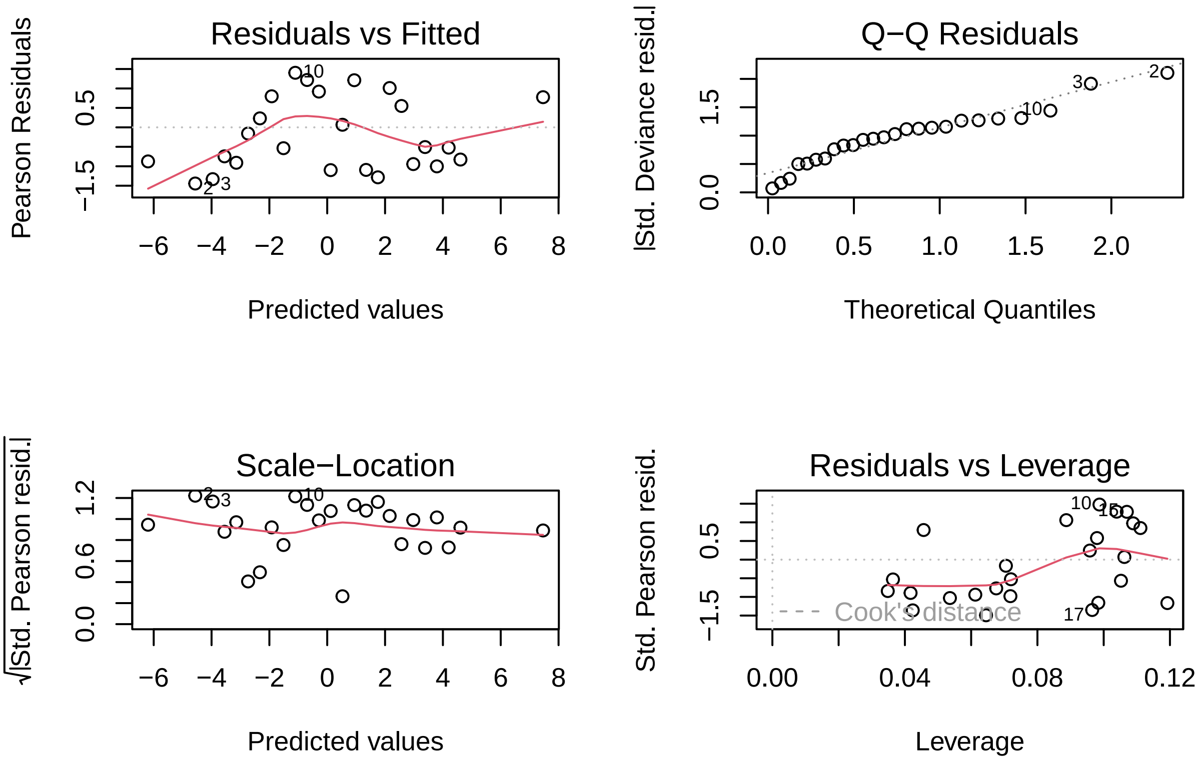
```
anova(m.men2.glm, test="Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                    24       3694
Age   1     3667        23         27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test statistic, *Deviance*, has a χ^2 distribution. Chi-squared distributions have a single parameter, *degrees of freedom*. Large values of this test statistic provide evidence away from the null hypothesis that (in our case) $b_1 = 0$. In the output we can see that the p-value given by $\Pr(\chi^2 > 3667.2)$ is ≈ 0 ($3667.2 = 3693.9 - 26.7$). Hence the result is highly significant and we should keep *Age* in the model.

Before looking at the diagnostic plots we note that the mean residual deviance, the *scaled deviance*, is $26.7/23 = 1.16$. Binomial and Poisson models both have true dispersion (mean deviance) values of 1, thus the fact that the estimate from the model is close to 1 is a good sign.

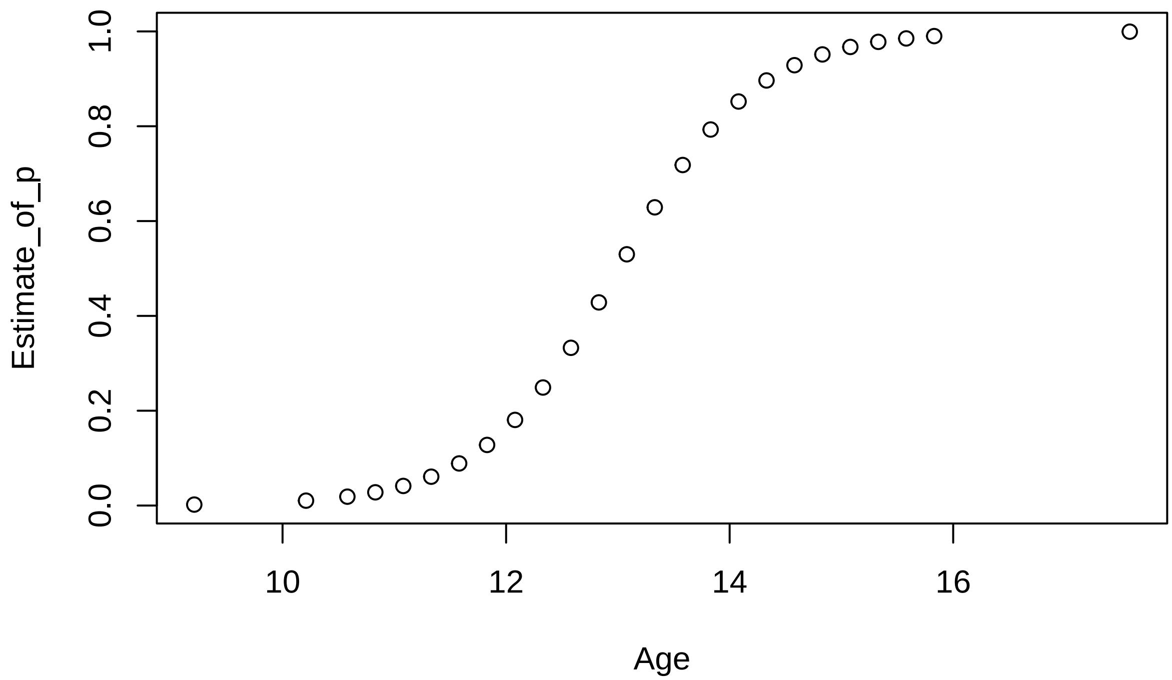
```
# Copy of current graphic parameters
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.men2.glm)
```




```r
# Revert to original settings
par(oldpar)
```

These plots show slightly improved residual behaviour over fitting an ordinary linear regression model to the logit transformed data as we did before. Another advantage of GLMs is that the fitted values are given in the natural scale for the response variable. So in this case in terms of p, as opposed to the transformed scale logit(p).

```r
Estimate_of_p <- m.men2.glm$fitted.values
with(menarche, plot(Age, Estimate_of_p))
```



Predictions (probabilities p in this case) for ages from 10 to 16 are easily obtained using `predict` as usually.

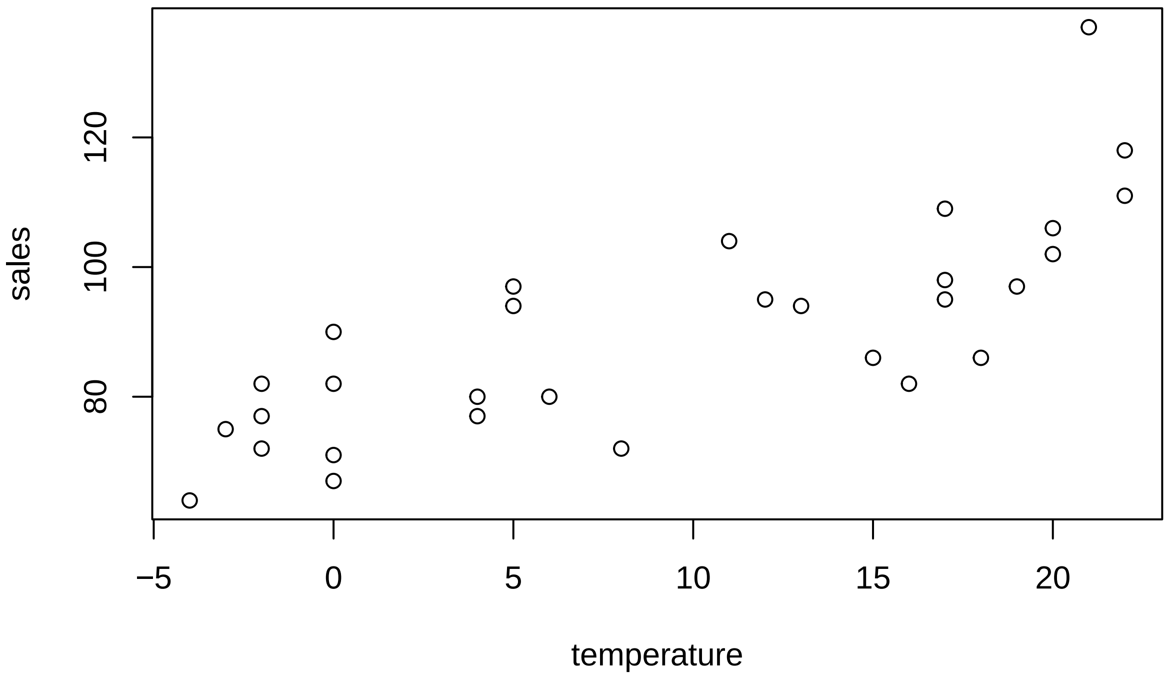
```
newvalues <- data.frame(Age=c(10,12,14,16))
predict(m.men2.glm,newvalues,type="response",se=TRUE)
```

```
$fit
        1         2         3         4 
0.0073425 0.1620879 0.8349553 0.9924983 

$se.fit
        1         2         3         4 
0.0013803 0.0118866 0.0117394 0.0013861 

$residual.scale
[1] 1
```

10.5 Poisson GLM regression

The *cans* data set refers to the number of cans sold in a vending machine each week, along with the average temperature for that week.

```
cans <- read.table("cans.txt",header=TRUE)
with(cans,plot(temperature, sales))
```



The sales data are counts and they are likely to follow a Poisson distribution. We want to see if mean weekly sales can be modelled as a function of temperature. The default link function for Poisson data is the natural logarithm (ln).

Letting Y denote the number of cans sold in a week, we model Y as a Poisson variable with mean λ as

$$Y \sim Poisson(\lambda)$$
$$\ln \lambda = b_0 + b_1 temperate$$

```
m.can.glm <- glm(sales ~ temperature, family=poisson,data=cans)
summary(m.can.glm)
```

```

Call:
glm(formula = sales ~ temperature, family = poisson, data = cans)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.34098    0.03025  143.49  < 2e-16 ***
temperature  0.01602    0.00222    7.22  5.1e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 84.661  on 29  degrees of freedom
Residual deviance: 32.047  on 28  degrees of freedom
AIC: 225.8

Number of Fisher Scoring iterations: 4
```

The fitted model involves a Poisson distribution for weekly sales with parameter λ where $\ln \lambda = 4.34 - 0.0160 temperature$. The t-test suggests temperature is strongly significant but again it is better to use the

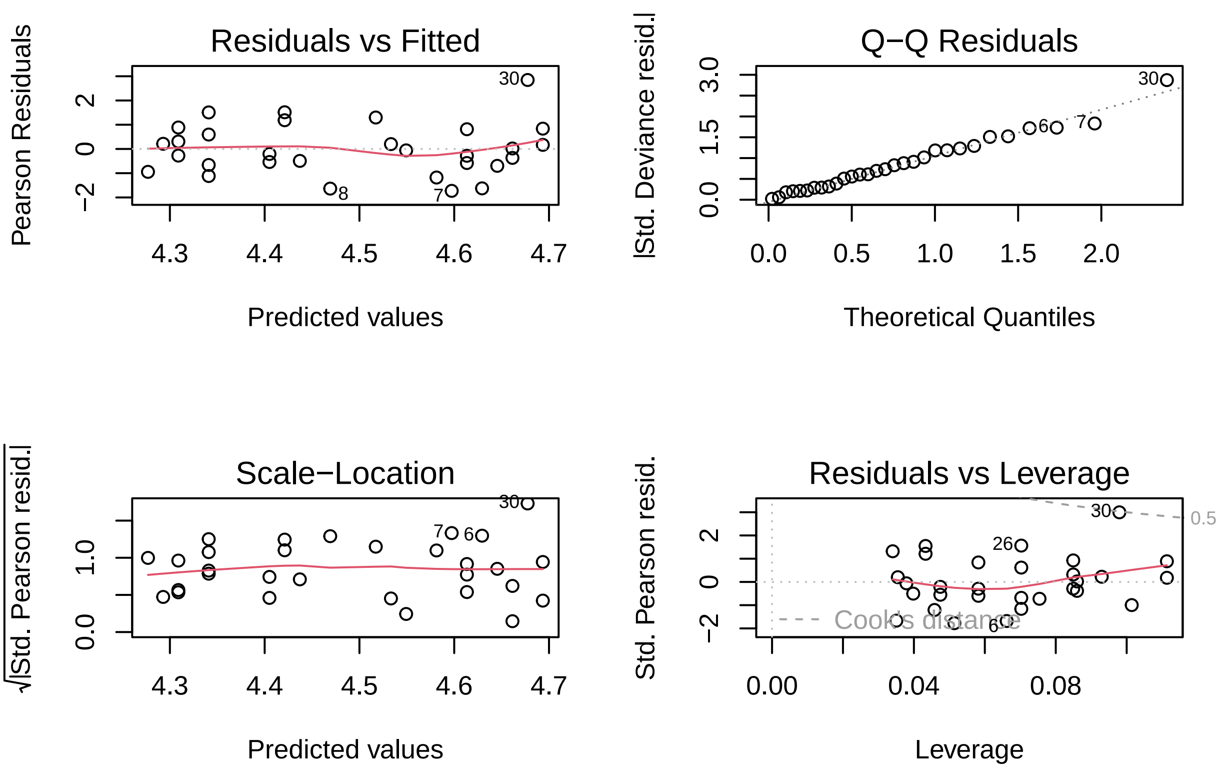
χ^2 test.

```r
anova(m.can.glm, test="Chisq")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: sales

Terms added sequentially (first to last)

            Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                           29       84.7
temperature  1     52.6        28       32.0  4.1e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the dispersion (mean residual deviance) is estimated as 32.05/28, approximately 1, the true value we would expect when the data follow a Poisson distribution. The resulting p-value is very small and hence suggests that *temperature* is associated with *sales*. The diagnostic plots indicate a reasonable fit.

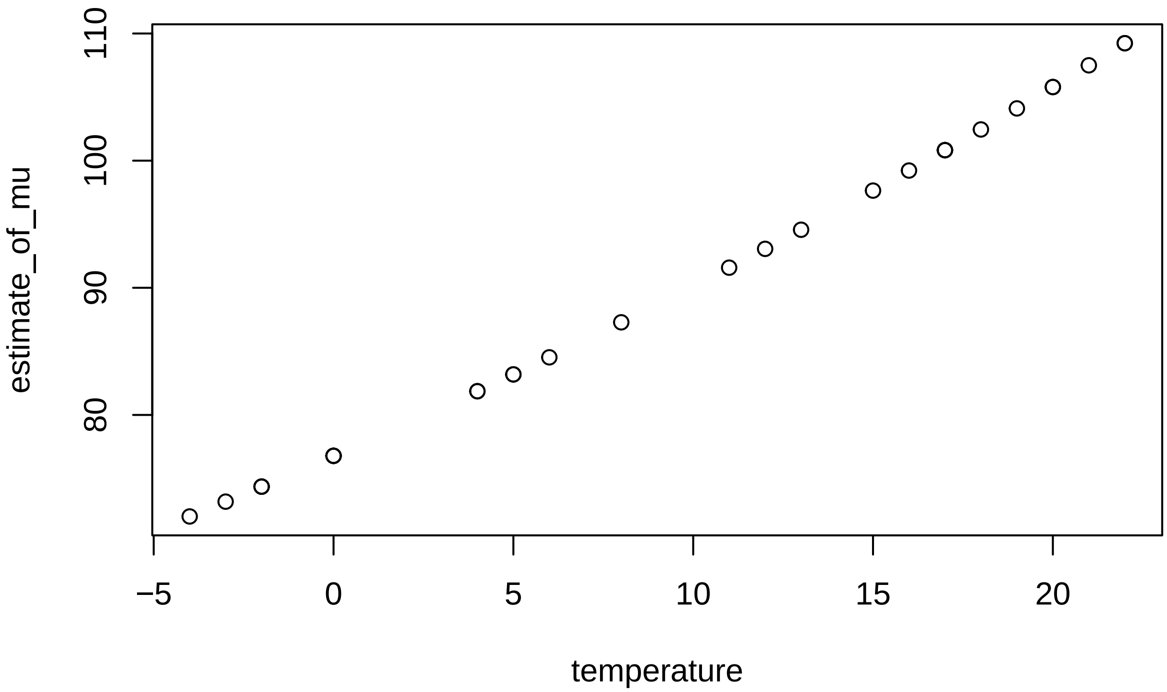
```r
# Copy of current graphic parameters
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.can.glm)
```




```r
# Revert to original settings
par(oldpar)
```

We can obtain estimated mean sales with temperature.

```
estimate_of_mu <- m.can.glm $fitted.values
with(cans, plot(temperature,estimate_of_mu))
```



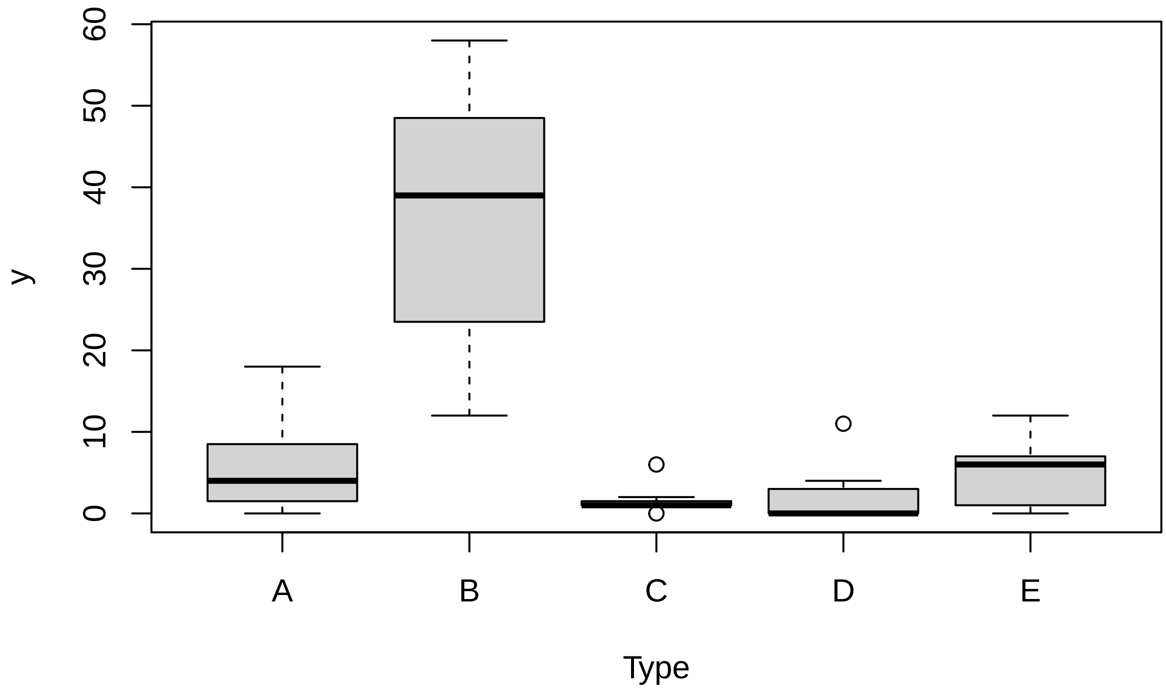
10.6 Explanatory factor variables, offsets and interaction effects

We extend GLMs to include categorical variables, which must be introduced as factors into the `glm` function. We also include an *offset*, that is a variable in the linear predictor whose coefficient we insist takes the value 1. An offset is used to account for a baseline value for the response variable to make the values of this latter comparable between individuals.

The data file `Damage.txt` contains the number of times waves caused damage to forward sections of cargo-carrying ships. To set standards of hull construction, the damage risk associated with various factors is assessed.

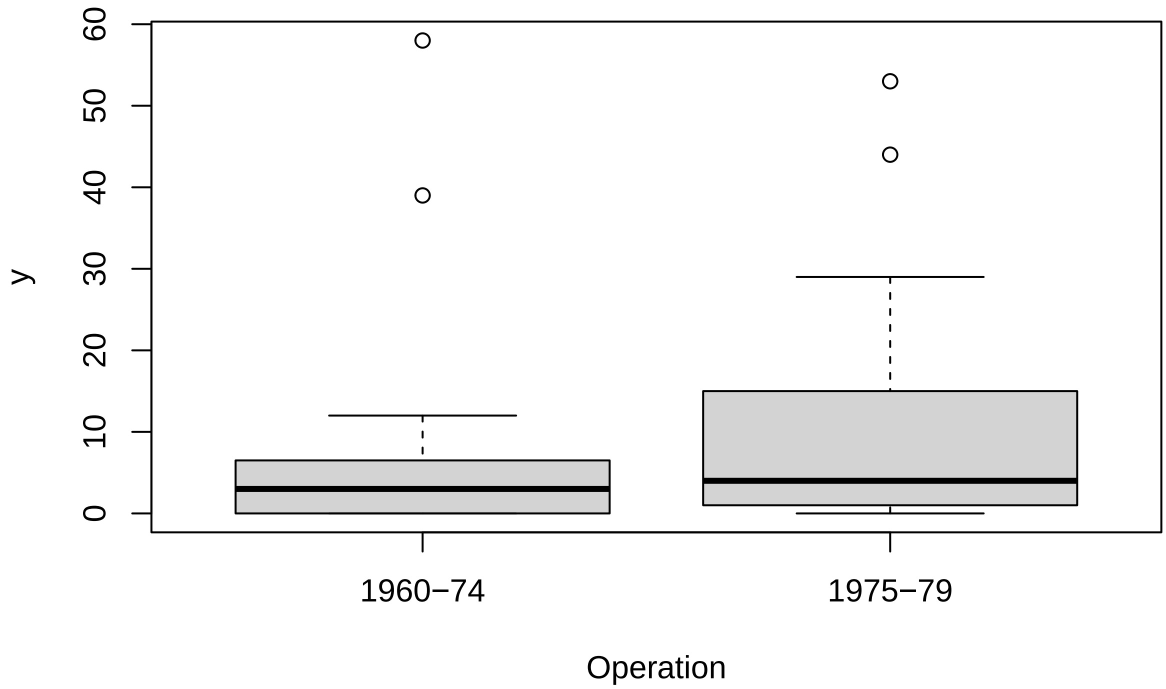
The following plots show wave damage in cargo ships by ship type, construction date, and time of occurrence.

```
ships <- read.table("ships.txt",header=TRUE, stringsAsFactors = TRUE)
plot(ships$Type, ships$Damage, xlab= "Type",)
```




```r
plot(ships$Construct, ships$Damage, xlab= "Construction")
```


```r
plot(ships$Operation, ships$Damage, xlab= "Operation")
```



Damage measures are counts and we begin by assuming they have a Poisson distribution. We want to know if the mean number of damage incidents is affected by *Type*, *Construct* or *Operation*, or some combination of these. Our first model is

$$\log(\lambda) = mean + Effect(Type) + Effect(Construct) + Effect(Operation)$$

However we also have data on how long each ship was in service, in months, and it would seem more sensible to model the mean number of incidents per month, i.e. to model $\log(\lambda/Service)$. But $\log(\lambda/Service) = \log\lambda - \log(Service)$. So the model becomes

$$\log(\lambda) = mean + log(Service) + Effect(Type) + Effect(Construct) + Effect(Operation)$$

We now have $\log(Service)$ in the model as an offset (its coefficient is fixed at 1).

```
ships$logS  <- log(ships$Service)
m.ship.glm <-
  glm(Damage ~ offset(logS) + Type + Construct + Operation,
      family = poisson,
      data = ships)
anova(m.ship.glm, test="Chisq")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: Damage

Terms added sequentially (first to last)


          Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                         33      146.3
Type       4     55.4        29       90.9  2.6e-11 ***
Construct  3     41.5        26       49.4  5.0e-09 ***
```

```
Operation  1     10.7        25       38.7   0.0011 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that all three factors, *Type*, *Construction* and *Operation* affect *Damage*. It can happen that the effects of a factor vary according to the levels of another factor (or more generally according to the values of another covariate). This is called an interaction effect and we can include this in the model, for example there is some evidence in the area about a *Type* by *Construction* interaction (this is a *2-way* interaction). Single 2-way interactions are specified with the : operator in R formulae.

```r
m.ship.glm.int <-
  glm(
    Damage ~ offset(logS) + Type + Construct + Operation + Type:Construct,
    family = poisson,
    data = ships
  )
anova(m.ship.glm.int, test="Chisq")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: Damage

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                              33      146.3
Type            4     55.4        29       90.9  2.6e-11 ***
Construct       3     41.5        26       49.4  5.0e-09 ***
Operation       1     10.7        25       38.7   0.0011 **
Type:Construct 12     24.1        13       14.6   0.0197 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A full model considering all main effects and possible interactions (2-way and 3-way in this case) is specified as follows.

```r
m.ship.glm.int2 <-
  glm(
    Damage ~ offset(logS) + Type*Construct*Operation,
    family = poisson,
    data = ships
  )
anova(m.ship.glm.int2, test="Chisq")
```

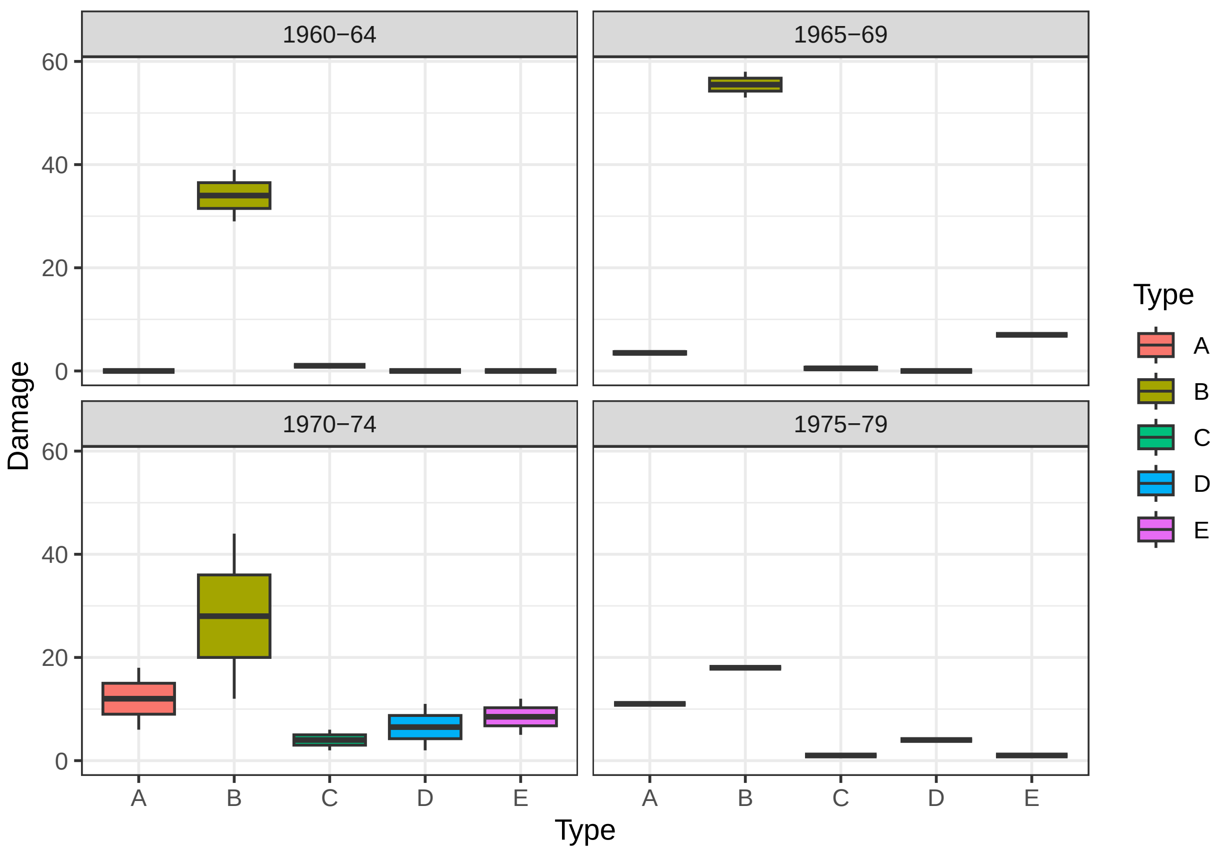
```
Analysis of Deviance Table

Model: poisson, link: log

Response: Damage

Terms added sequentially (first to last)


                         Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                                        33      146.3
Type                      4     55.4        29       90.9  2.6e-11 ***
```

```
Construct                3     41.5        26       49.4  5.0e-09 ***
Operation                1     10.7        25       38.7   0.0011 **
Type:Construct          12     24.1        13       14.6   0.0197 *
Type:Operation           4      6.1         9        8.5   0.1943
Construct:Operation      2      1.7         7        6.9   0.4351
Type:Construct:Operation 7      6.9         0        0.0   0.4440
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Along with the main effects of the factors, following the usual statistical significance criteria, the only significant interaction is that between *Type* by *Construction*. So the model could be simplified to `m.ship.glm.int` above.

In fact, the following exploratory plots of the data (produced using the package `ggplot2`) would have already suggested this potential interaction effect, the effect of *Type* on *Damage* appears to change depending on *Construct*.

```
library(ggplot2)
ggplot(aes(x=Type, y=Damage, fill = Type), data = ships) +
  facet_wrap(~Construct) +
  geom_boxplot() +
  theme_bw()
```



We have illustrated the inclusion of interactions here, but this applies to any regression and ANOVA model. However, note that including too many interactions can lead to fitting problems and complicates the interpretation of the models. So their specification should be better guided by sensible considerations in the context of application.

Finally note that we have illustrated the use of GLMs with gaussian, binomial and Poisson data distributions, but other families can be similarly specified though the `family` argument of the `glm` function (see `?family`).

10.7 Exercises

1. The data file `pigs.txt` contains the number of occurrences of various types of behaviour among pigs in pens. Not all pens had the same number of pigs and only totals were recorded as it was not possible to observe each pig individually. The pens either had straw or did not have straw. Prior to the experiment the pigs had been housed in three different mixing treatments which related to what extent the animals remained with their siblings and the size of group they were living in. Using a Poisson distribution to model the counts, with an offset for the number of pigs in the pen, and a log link, decide whether straw or mixing or an interaction between straw and mixing affected (a) the feeding counts (b) the drinking counts (c) the 'up' counts. Check that your final models have well behaved residuals.
2. In the file `drop.txt` there are data about 36 batches of 8 potato tubers. These were dropped from various heights onto bars to test their resistance to splitting. How do tuber weight and drop height affect the probability of a tuber splitting?

11 Additive models

11.1 Introduction

We have already met smooth splines in the context of a single X variable, where we fit a smooth curve through the data points. We now extend that procedure to several X variables. For multiple regression we had

$$Y = b_0 + b_1 X_1 + ... + b_p X_p + \epsilon$$

Additive models instead consider smooth splines f for the explanatory variables

$$Y = b_0 + b_1 f_1(X_1) + ... + b_p f_p(X_p) + \epsilon$$

This provides a great flexibility to fit non-linear relationships between explanatory and response variables.

11.2 GAM analysis of the ozone data set

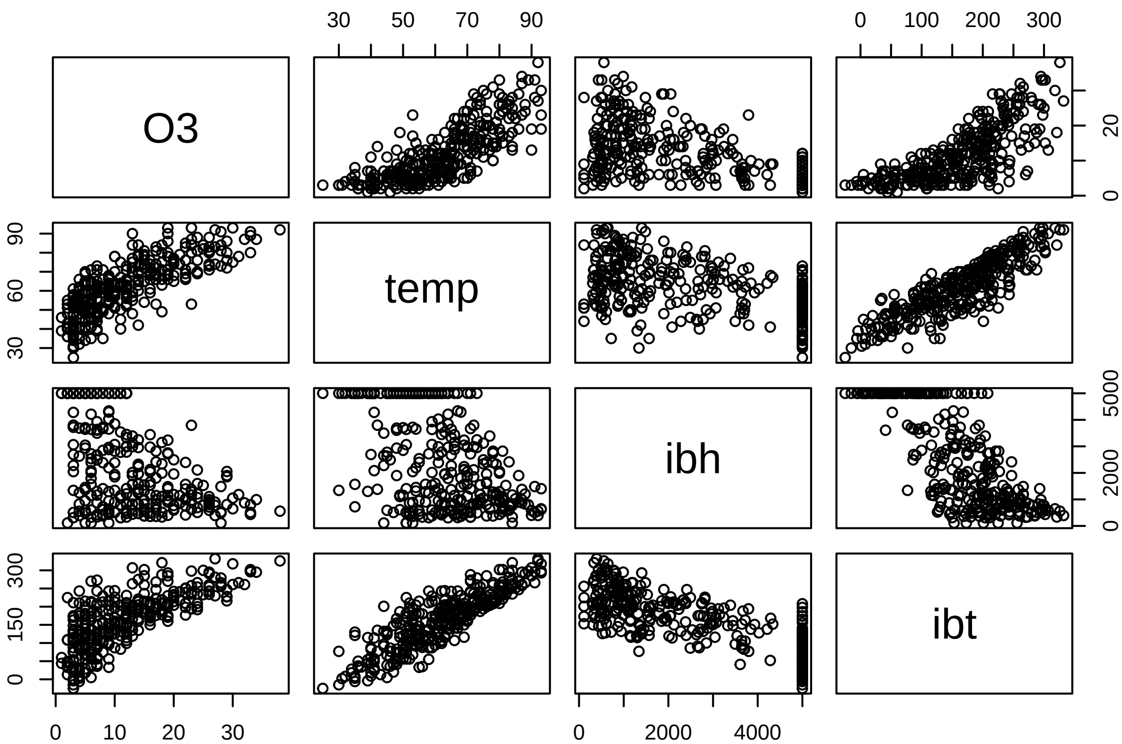
The package `mgcv`, accompanying the book *Generalized Additive Models* by Simon R. Wood, is the main reference to fit additive models in R. We will use it along with the ozone data set from the package `faraway`.

The ozone data are from a study in the Los Angeles Basin in 1976 to examine the relationship between ozone (O_3) and other meteorological variables. We will use only temperature (*temp*), inversion base height (*ibh*) and inversion top temperature (*ibt*).

```
# install.packages("faraway")
library(faraway)
str(ozone)
```

```
'data.frame':	330 obs. of  10 variables:
 $ O3      : num  3 5 5 6 4 4 6 7 4 6 ...
 $ vh      : num  5710 5700 5760 5720 5790 5790 5700 5700 5770 5720 ...
 $ wind    : num  4 3 3 4 6 3 3 3 8 3 ...
 $ humidity: num  28 37 51 69 19 25 73 59 27 44 ...
 $ temp    : num  40 45 54 35 45 55 41 44 54 51 ...
 $ ibh     : num  2693 590 1450 1568 2631 ...
 $ dpg     : num  -25 -24 25 15 -33 -28 23 -2 -19 9 ...
 $ ibt     : num  87 128 139 121 123 182 114 91 92 173 ...
 $ vis     : num  250 100 60 60 100 250 120 120 120 150 ...
 $ doy     : num  33 34 35 36 37 38 39 40 41 42 ...
```

```r
with(ozone, pairs(O3 ~ temp + ibh + ibt))
```



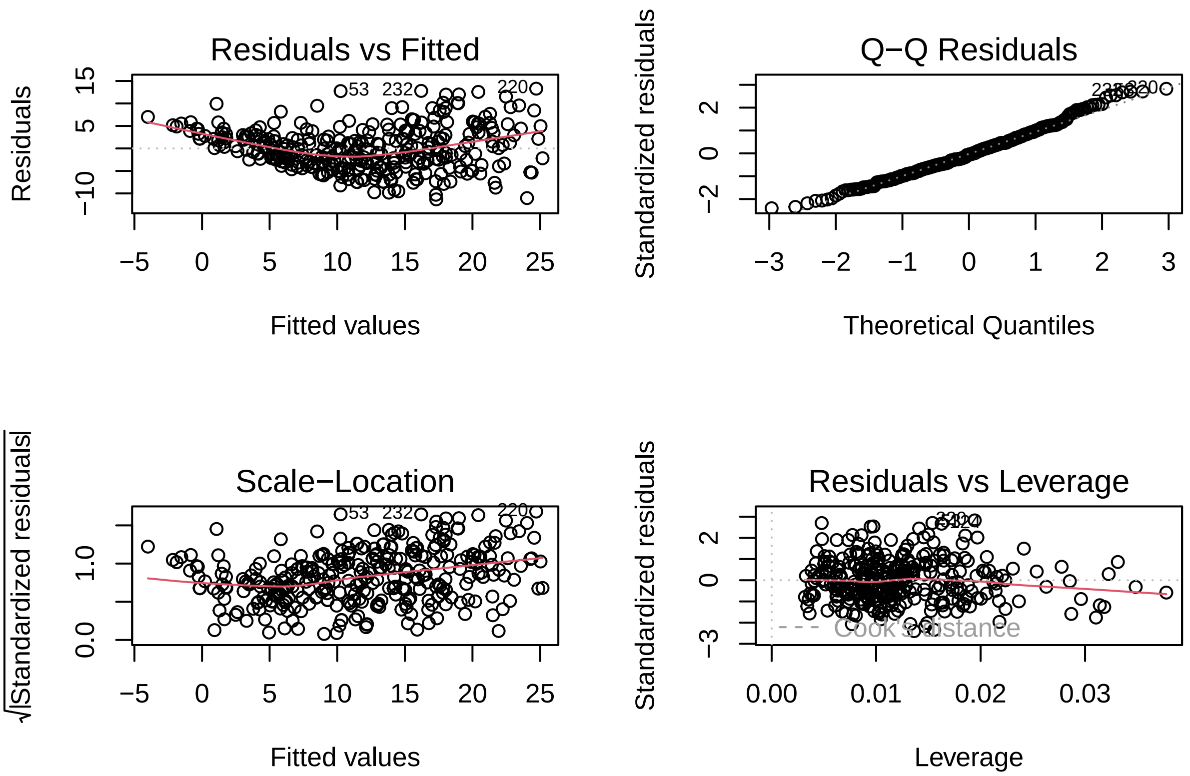
The pairs plot shows O_3 responding non-linearly to *temp* and *ibt*, but we fit a simple linear model as a starting point.

```r
m.olm <- lm(O3 ~ temp+ibh+ibt, data=ozone)
summary(m.olm)
```

```

Call:
lm(formula = O3 ~ temp + ibh + ibt, data = ozone)

Residuals:
    Min      1Q  Median      3Q     Max 
-11.322  -3.191  -0.259   2.963  13.286 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -7.727982   1.621662   -4.77  2.8e-06 ***
temp         0.380441   0.040158    9.47  < 2e-16 ***
ibh         -0.001186   0.000257   -4.62  5.5e-06 ***
ibt         -0.005821   0.010179   -0.57     0.57    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.75 on 326 degrees of freedom
Multiple R-squared:  0.652,	Adjusted R-squared:  0.649 
F-statistic:  204 on 3 and 326 DF,  p-value: <2e-16
```

```r
# Copy of current graphic parameters
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
```

```r
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.olm)
```




```r
# Revert to original settings
par(oldpar)
```

The residuals-versus-fitted plot indicates a problem of increasing variance as the mean of O_3 increases and the q-q plot has a problem in its upper tail. We need to try something else.

In the following we use the function `gam` of the package `mgcv` to fit a generalised additive model (GAM) assuming a Gaussian response variable.

```r
# install.packages("mgcv")
library(mgcv)
m.am <- gam(O3 ~ s(temp) + s(ibh) + s(ibt), data=ozone)
summary(m.am)
```

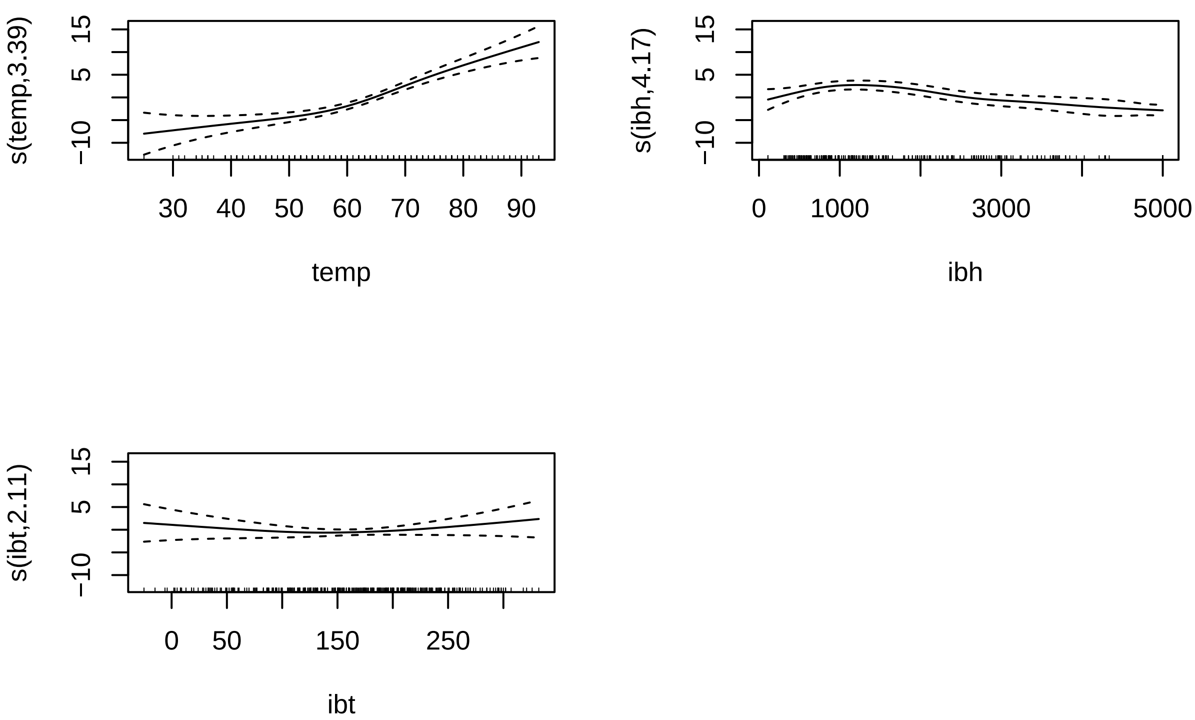
```
Family: gaussian
Link function: identity

Formula:
O3 ~ s(temp) + s(ibh) + s(ibt)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   11.776      0.238    49.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
```

```
          edf Ref.df     F p-value
s(temp) 3.39   4.26 20.68 < 2e-16 ***
s(ibh)  4.17   5.08  7.34 1.7e-06 ***
s(ibt)  2.11   2.73  1.40    0.21
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.708   Deviance explained = 71.7%
GCV = 19.346  Scale est. = 18.72     n = 330
```

This has done a little better than the ordinary linear model judging by the adjusted R^2, but has used more degrees of freedom (edf in the R output). Again *ibt* seems not to affect O_3. Note that `gam` has automatically determined the amount of smoothing. This is done by minimizing a cross-validation measure, the GCV score.

We can see the shape of the fitted smooth functions easily.

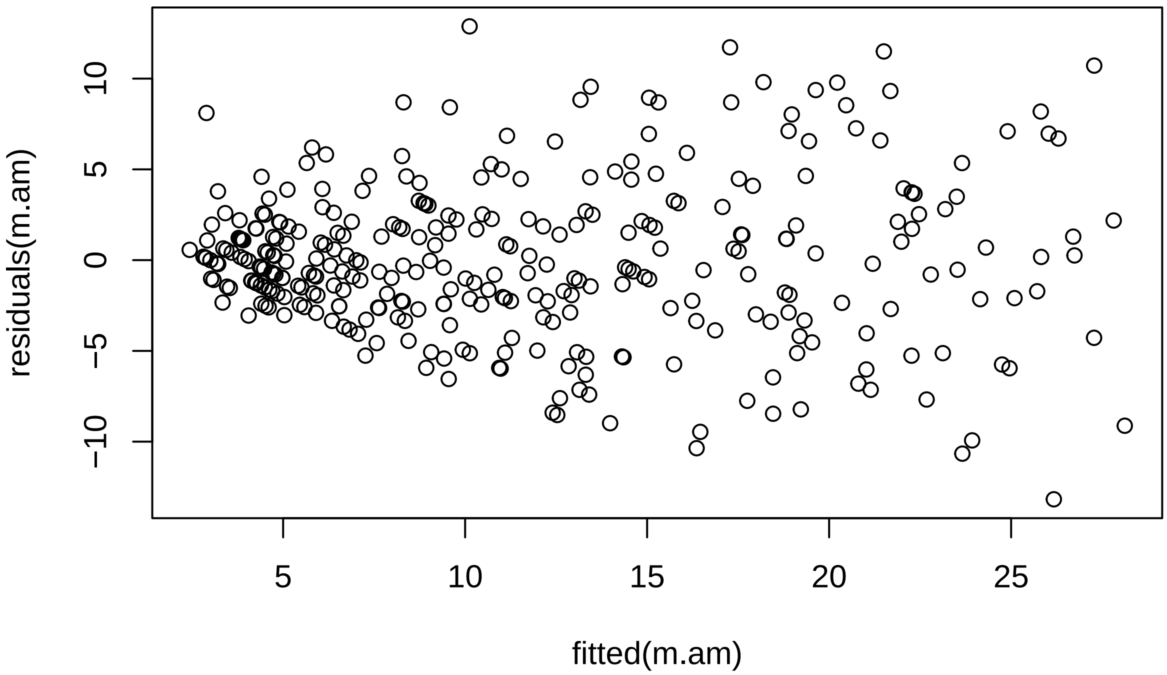
```r
# Copy of current graphic parameters
oldpar <- par()
# Modify them to do a 2x2 matrix of graphs
par(oma = c(0, 0, 0, 0), mfrow=c(2, 2))
plot(m.am)
# Revert to original settings
par(oldpar)
```



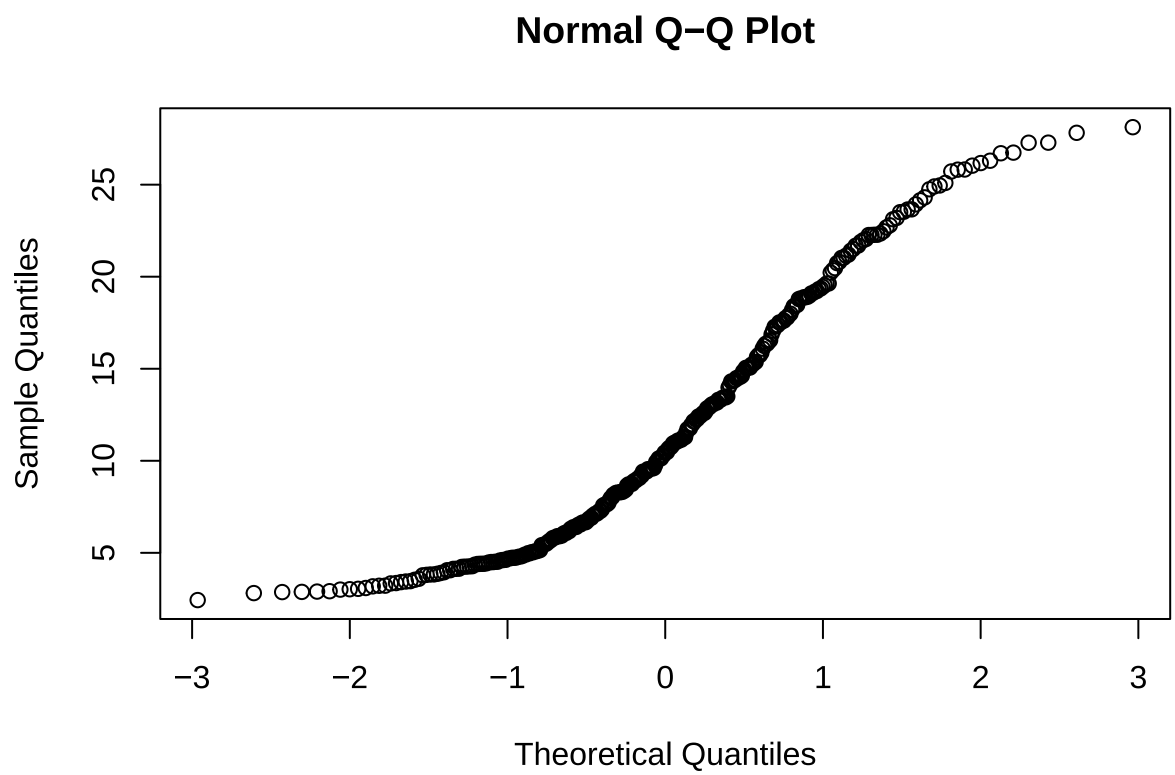
These results suggest that the effect of *ibt* on O_3 is minimal, that *ibh* has more effect but is highly variable and that *temp* has a strong positive link with O_3 but that it might get stronger above 60 degrees.

The diagnostic plots have improved a little with respect to the linear model but not enough.

```r
plot(fitted(m.am), residuals(m.am))
```




```r
qqnorm(fitted(m.am))
```



Next we drop *ibt* because it was not significant.

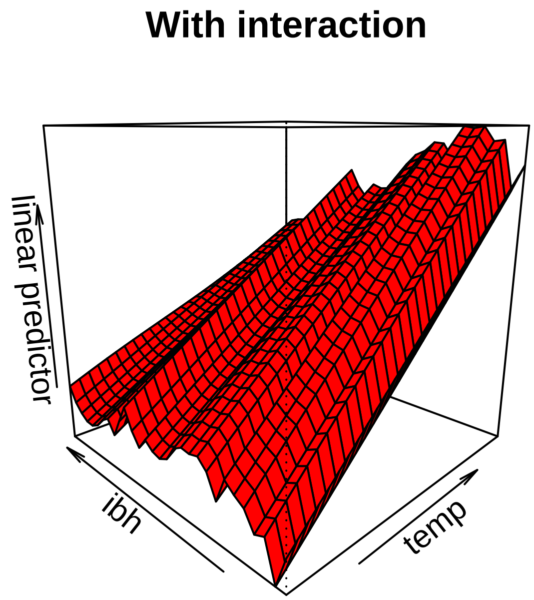
```r
m.am2 <- gam(O3 ~ s(temp) + s(ibh), data=ozone)
```

We also try a possible interaction between *temp* and *ibh*,

```r
m.amint <- gam(O3 ~ s(temp,ibh), data=ozone)
```

and note that the adjusted R^2 has hardly changed from the model without the interaction term. It is easy to visualise the predictions from the model with interaction using `vis.gam`.

```r
vis.gam(m.amint, view=c("temp","ibh"), se=0, theta=-45, col="red", main="With interaction")
```



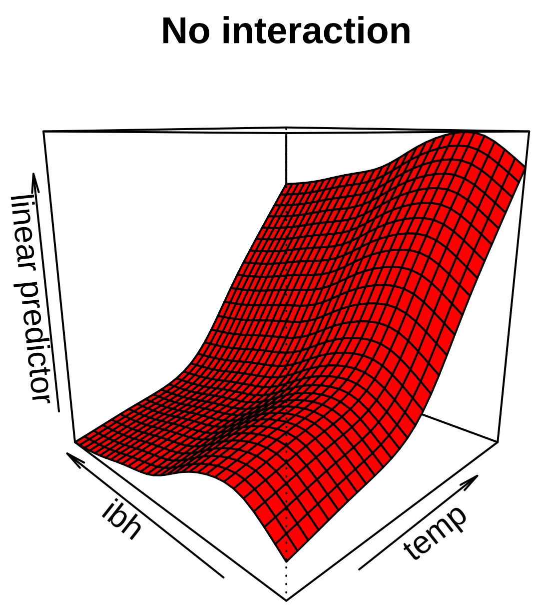
We can compare more formally models `m.amint` and `m.am2` (with and without interaction) to confirm our conclusion that there is no need to allow for interaction.

```r
anova(m.am2,m.amint, test="F")
```

```
Analysis of Deviance Table

Model 1: O3 ~ s(temp) + s(ibh)
Model 2: O3 ~ s(temp, ibh)
  Resid. Df Resid. Dev   Df Deviance    F Pr(>F)
1       319       6054                          
2       302       5896 17.2      158 0.48   0.96
```

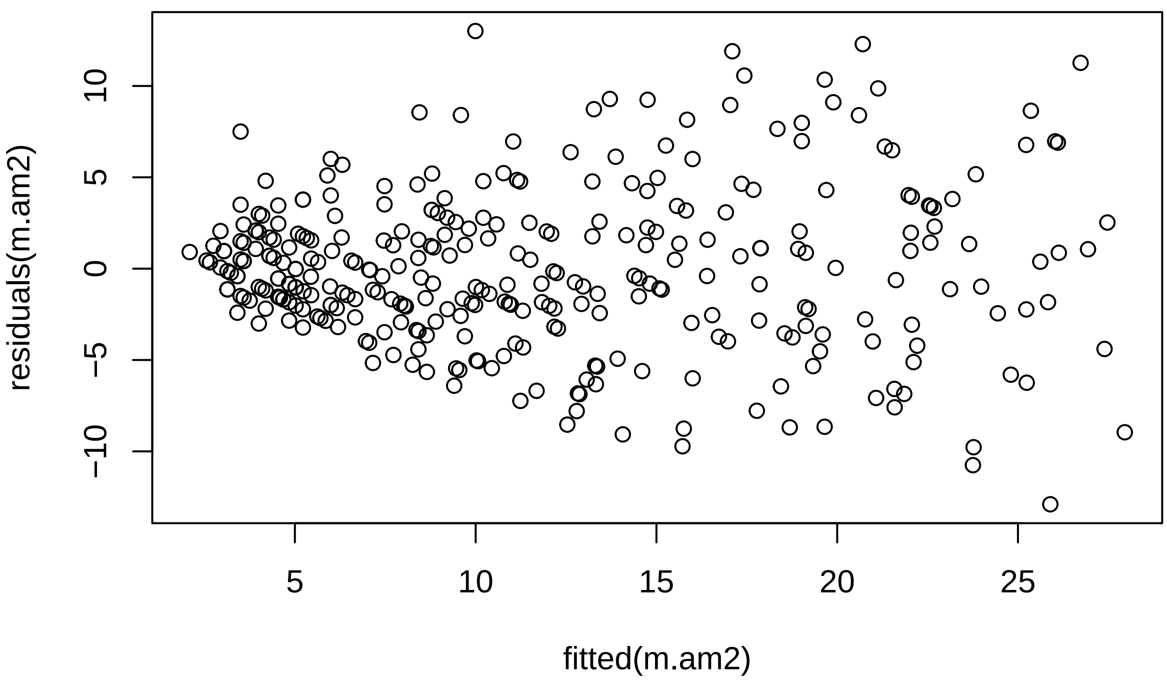
The F-test suggests that the models are actually statistically comparable and it is not worth including the interaction term.

```r
vis.gam(m.am2, view=c("temp","ibh"), theta=-45, col="red", main="No interaction")
```

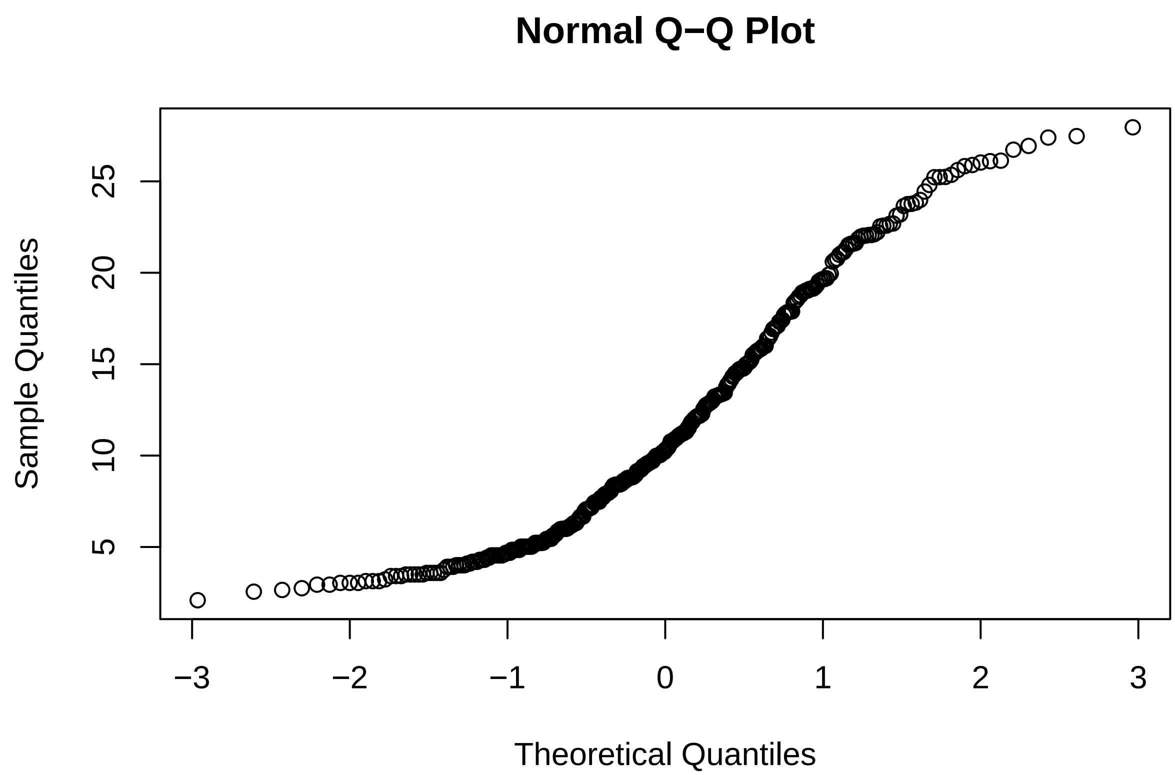


The diagnostic plots for model `m.am2` still leave room for improvement.

```r
plot(fitted(m.am2), residuals(m.am2))
```




```r
qqnorm(fitted(m.am2))
```



While smooth functions do not provide mathematical formulae or parameters by which we can characterise a process, they are sometimes used as a first step in determining an appropriate parametric model.

Alternatively, the plots for model `m.am` hint that both *temp* and *ibh* might affect O_3 in a 'broken stick' manner. The response of O_3 might be a function of two different straight lines with the break point at $temp = 60$; similarly for *ibh* with a break point at $ibh = 1000$. We can explore this possibility considering the following linear model.

```
rhs <- function(x,c) ifelse(x>c,x-c,0)
lhs <- function(x,c) ifelse(x<c,c-x,0)
m.olm2 <-
  lm(O3 ~ rhs(temp, 60) +
          lhs(temp, 60) +
          rhs(ibh, 1000) +
          lhs(ibh, 1000),
          data = ozone)
summary(m.olm2)
```

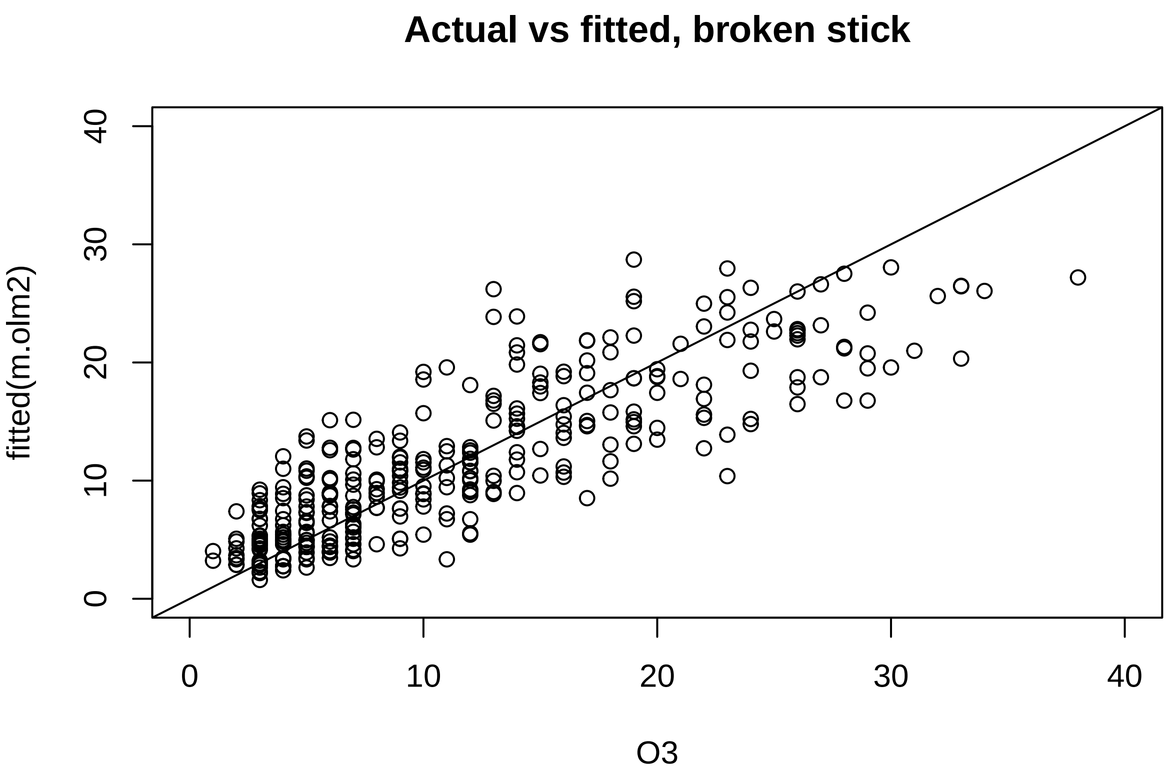
```

Call:
lm(formula = O3 ~ rhs(temp, 60) + lhs(temp, 60) + rhs(ibh, 1000) +
    lhs(ibh, 1000), data = ozone)

Residuals:
    Min      1Q  Median      3Q     Max
-13.204  -2.631  -0.289   2.318  12.672

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    11.603832   0.622651   18.64  < 2e-16 ***
rhs(temp, 60)   0.536441   0.033185   16.17  < 2e-16 ***
```

```
lhs(temp, 60)  -0.116173   0.037866   -3.07   0.0023 **
rhs(ibh, 1000) -0.001486   0.000198   -7.49  6.7e-13 ***
lhs(ibh, 1000) -0.003554   0.001314   -2.71   0.0072 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.34 on 325 degrees of freedom
Multiple R-squared:  0.71,  Adjusted R-squared:  0.706
F-statistic:  199 on 4 and 325 DF,  p-value: <2e-16
```

Adjusted R^2, residual standard error and diagnostic plots (not shown) all indicate the same degree of goodness of fit and the same problems as the `m.am2` model. What we have gained is an explicit model with parameters we can use to compare directly with other data sets.

To see the weakness of the `m.olm2` model more clearly we plot its fitted values against the actual O_3 values.

```
with(ozone, plot(O3,fitted(m.olm2), ylim=c(0,40), xlim=c(0,40),
                 main="Actual vs fitted, broken stick") )
abline(a=0,b=1)
```



Using the ozone data set we have illustrated the use of additive models with a Gaussian response variable. However, sticking with the package `mgcv` and the same function `gam` we can easily extend this to generalized additive models (GAMs) through the `family` argument. This allows to use smooth functions to fit models with no Gaussian responses in a form equivalent to fitting generalised linear models using `glm` as we discussed before.

11.3 Exercises

1. One of the weaknesses with all of the ozone additive models we have tried is the increasing variance in the residuals as the mean value of O_3 increased. Try modelling O_3 as following a Gamma distribution

with identity link instead of the default Gaussian distribution. This will allow for increasing variance as O_3 increases. Comment the results in comparison with the previous models fitted.

2. With the savings data is there any merit in using smooth functions rather than linear functions to predict sr?

12 Regression analysis with many variables

12.1 Introduction

The standard regression model consists of fitting an equation of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

to a data set including a response/dependent variable y and a number of explanatory/independent variables $x_1, \ldots, x_p$. However, when faced with many explanatory variables or more variables than samples in the data set, straightforward multiple regression struggles or simply fails. In particular the standard model is unsatisfactory when:

- p is large relative to n,
- the explanatory variables x_i are highly correlated,
- and directly fails when $p \geq n$.

We here briefly describe and illustrate two popular approaches to get around these difficulties:

- Principal Component Regression (PCR).
- Partial Least Squares (PLS).

Not surprisingly at this point of the course, the idea consists of projecting the data onto low dimensions.

12.2 PC regression of the NIR data set

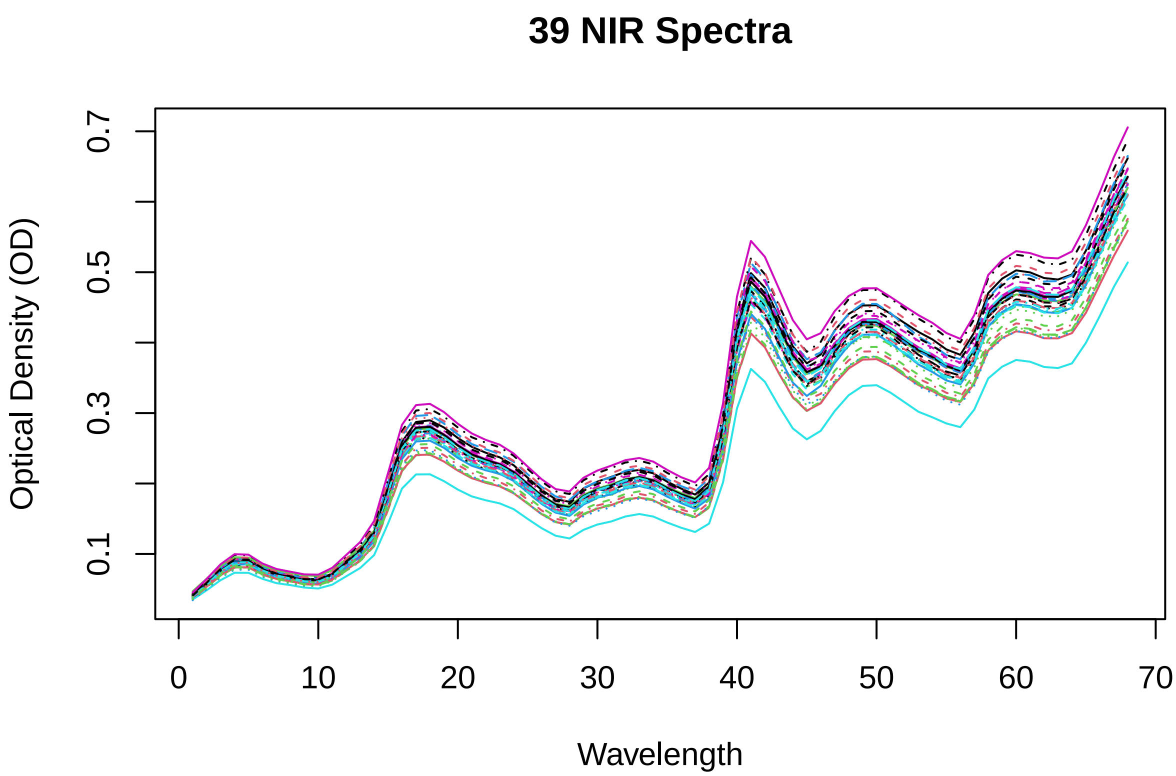
The data set consists of near-infrared reflectance (NIR) spectra from 39 samples of wheat flour which account for the intensity of reflected light measured (optical density, OD) at 68 wavelengths (in increasing order). Additionally, sample protein content was assessed by wet chemistry.

```
nir <- read.table("nir.txt", h = T)
```

Protein content and moisture are in the first columns and the NIR spectra along wavelengths are in the remaining columns. The objective is to fit a regression model to predict protein content from NIR spectra. The problem is that ordinary regression (say using the `lm` function in R) will fail because the data matrix has more columns than rows. Even if we had 80 or 90 samples there are still some additional estimation problems. Firstly, how do we select the important wavelengths from among so many? Secondly, if we select many wavelengths there is a strong chance of over-fitting. Finally, the ODs are highly correlated with each other (you can check this out using `cor` on a subset of them), which makes variable selection even more difficult.

Before proceeding we obtain a plot of the NIR spectra. The data values are columns 3 to 70 of the `data.frame`.

```
wv <- 1:68 # Range of wavelengths
# Extract spectra ODs and transpose to arrange them by rows (required by matplot)
OD <- t(nir[, 3:70])
matplot(wv, OD, type = "l", xlab = "Wavelength", ylab = "Optical Density (OD)",
  main = "39 NIR Spectra")
```



We can apply PCA on the explanatory variables $\mathbf{X}$ and fit a regression equation of the form

$$\text{Protein} = \beta_0 + \beta_{(1)}PC_{(1)} + \beta_{(2)}PC_{(2)} + ... + \beta_{(k)}PC_{(k)}$$

where $PC_{(i)}$ refers to the i^{th} selected principal component from $\mathbf{X}$.

But which PCs we choose? It is a good idea to select those which have the strongest correlation with the response variable, protein in this case. Note that these are not necessarily the first PCs (those explaining most variability in $\mathbf{X}$).

We derive the PCs and select those which have the strongest correlation with protein.

```
# Principal component analysis
nir.pca <- prcomp(nir[, 3:70])
# Consider e.g. the first 8 PCs
scores <- predict(nir.pca)[, 1:8]
# Extract loadings from PCA object
loadings <- nir.pca$rotation
```

The following computes the correlation between the response variables and the PCs.

```
round(cor(nir[, 1:2], scores),4)
```

```
             PC1     PC2     PC3     PC4     PC5     PC6     PC7    PC8
protein  0.6988  0.0925 -0.1891  0.6677 -0.1079 -0.0095  0.0280 0.0018
moisture 0.1634 -0.9735 -0.0908 -0.0444 -0.0625 -0.0216 -0.0092 0.0385
```

Notice how it is only the first, third perhaps, and fourth PC which are correlated with protein. This is interesting given the % variance accounted for by the PCs (check for this using `summary` on `nir.pca`).

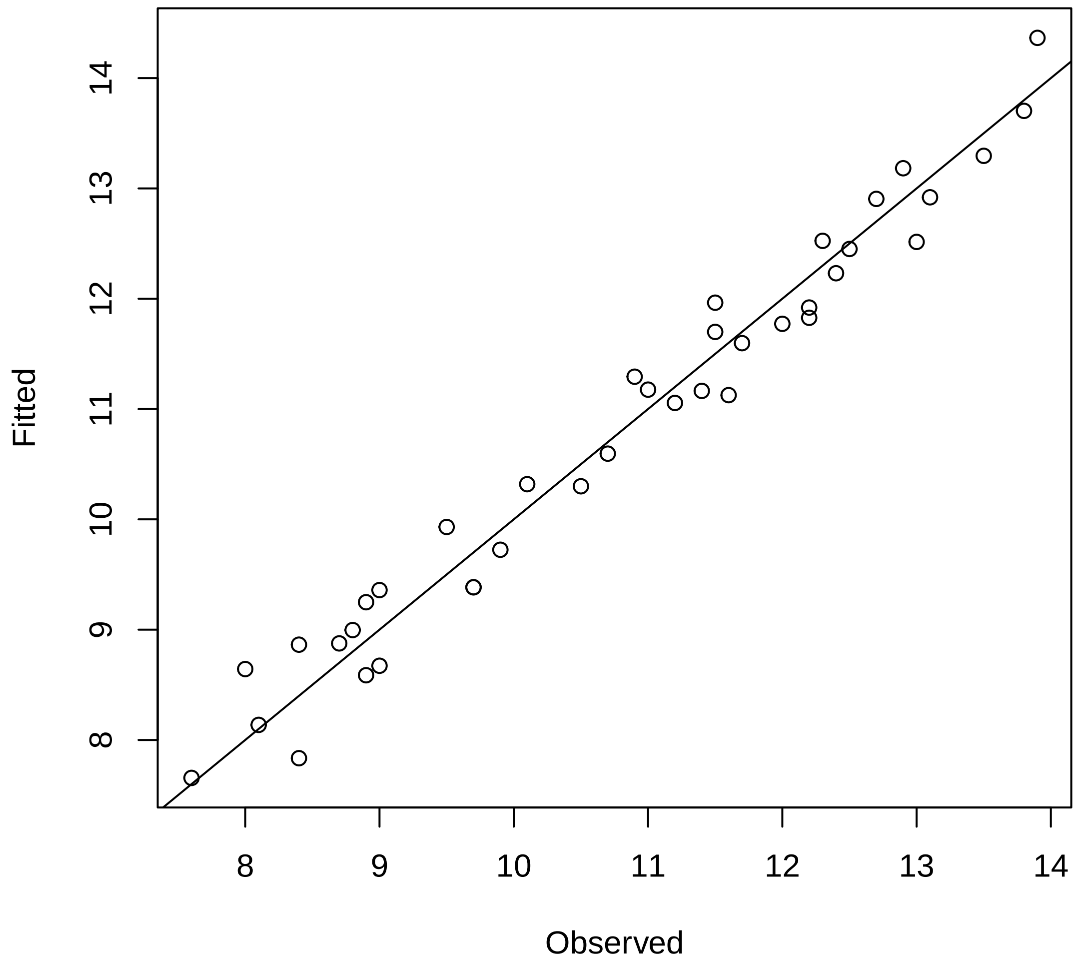
We now fit ordinary regression of protein on the selected PCs (1^{st}, 3^{rd} and 4^{th} PCs).

```
pcr <- lm(nir[, 1] ~ scores[, c(1, 3, 4)])
summary(pcr)
```

```
Call:
lm(formula = nir[, 1] ~ scores[, c(1, 3, 4)])

Residuals:
   Min     1Q Median     3Q    Max 
-0.643 -0.222  0.097  0.232  0.565 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)              10.8000     0.0524  206.00   <2e-16 ***
scores[, c(1, 3, 4)]PC1   6.8998     0.2896   23.83   <2e-16 ***
scores[, c(1, 3, 4)]PC3 -39.6930     6.1573   -6.45    2e-07 ***
scores[, c(1, 3, 4)]PC4 192.4147     8.4517   22.77   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.327 on 35 degrees of freedom
Multiple R-squared:  0.97,  Adjusted R-squared:  0.967 
F-statistic:  376 on 3 and 35 DF,  p-value: <2e-16
```

How good is the fit? We can look at the R^2 of the model and plot fitted versus observed protein values.

```r
plot(nir[, 1], pcr$fitted.values, xlab = "Observed", ylab = "Fitted")
abline(0, 1) # Add reference line
```



The coefficient of determination $R^2 = 0.9699$ is very high and the plot of observed versus fitted protein values shows high agreement. Hence, measuring NIR spectra and processing them through PCR seems to be a reliably approach to predict protein content in wheat flour as an alternative to direct measurement.

It is also possible to express this regression in terms of the original wavelengths. We have

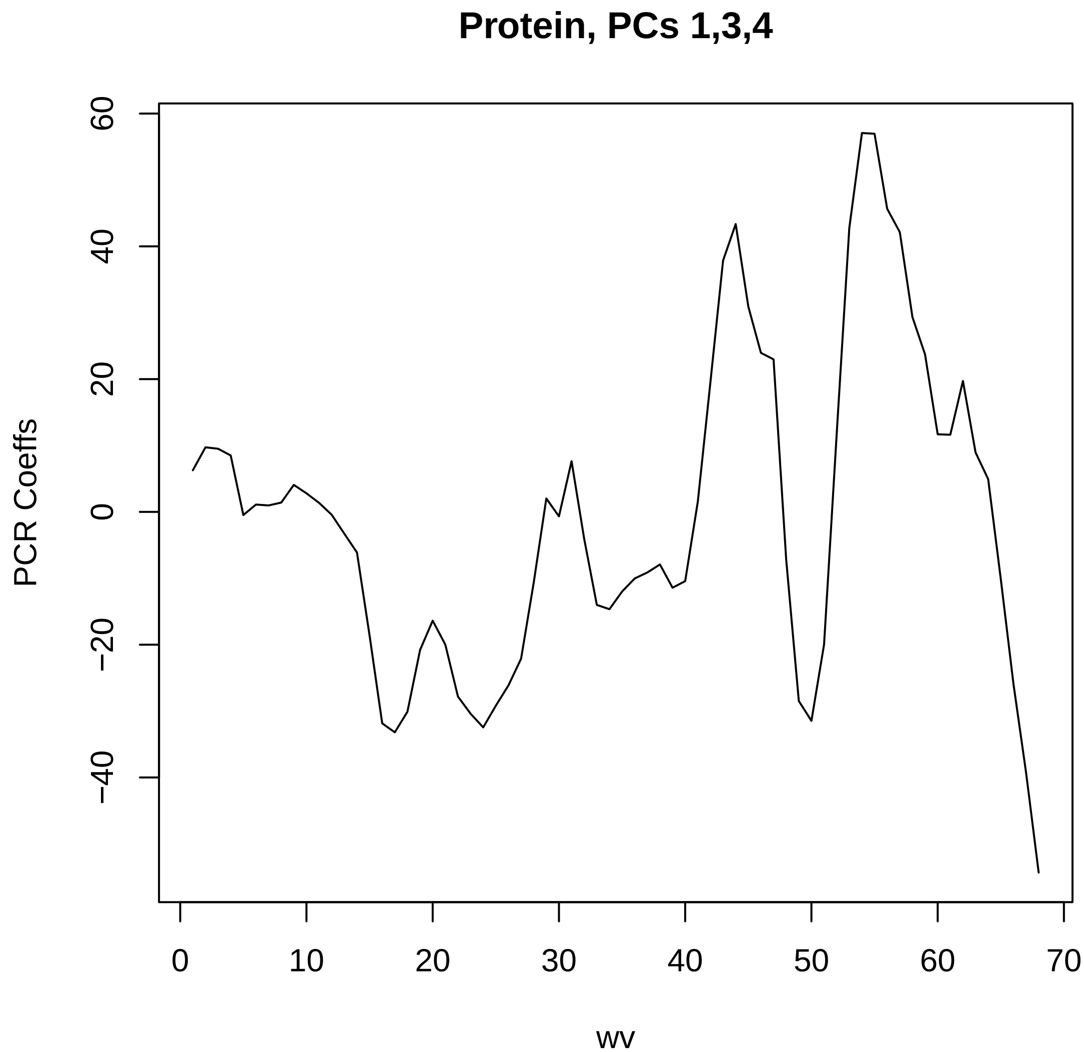
$$\text{Protein} = \beta_0 + \beta_{(1)} PC_{(1)} + ... + \beta_{(4)} PC_{(4)}$$

where each PC can be expressed as $PC_{(i)} = l_{1(i)} x_1 + ... + l_{68(i)} x_{68}$. We can simply substitute and gather together the terms for each wavelength x to express the PCR models as

$$\text{Protein} = \beta_0 + \delta_1 x_1 + ... + \delta_{68} x_{68}$$

A plot of these coefficients illustrates the relative weight of each wavelength for the prediction of protein.

```
beta <- pcr$coefficients
pcrmodel <- beta[2] * loadings[, 1] + beta[3] * loadings[, 3] + beta[4] * loadings[, 4]
plot(wv, pcrmodel, type = "l", ylab = "PCR Coeffs", main = "Protein, PCs 1,3,4")
```



12.3 PLS regression of the NIR data set

Partial least squares is an alternative to principal component regression and is very popular in the field of chemometrics. PLS addresses the same problem as PCR. Given a large number of independent variables x_i, often highly correlated, the purpose is to find a small number of linear combinations of the independent variables which are best predictors of the dependent variable y.

There are few computational formulations of the PLS algorithm but, in brief, PLS aims to find combinations

$$\boldsymbol{\lambda}'\mathbf{x} = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_p x_p$$

which maximise

$$\text{Corr}^2(y, \boldsymbol{\lambda}'\mathbf{x})\text{Var}(\boldsymbol{\lambda}'\mathbf{x}).$$

Some distinctive features are:

- PLS uses both the values of y and the x_i to find the latent components (often called factors).
- PCR uses $\mathbf{X}$ only to find the new components (PCs), then makes the link to y by ordinary regression.
- PLS factors and scores are interpreted as for PCs (PLS factors also uncorrelated).

- For regression, PLS selects the first k PLS factors based on model performance (R^2 and root mean squared error, RMSE, commonly).
- PLS generally uses 1 or 2 fewer terms than PCR to reach comparable goodness of fit.

We use the `pls` package to fit a PLS model to the NIR data set analogous to the previous one using PCR.

```
library(pls)
wv <- data.frame(nir[, 3:70])
pls <- plsr(nir$protein ~ ., ncomp = 6,data = wv, validation = "CV")
summary(pls)
```

```
Data:   X dimension: 39 68
    Y dimension: 39 1
Fit method: kernelpls
Number of components considered: 6

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           1.835    1.321   0.9242   0.3775    0.238   0.2159   0.2357
adjCV        1.835    1.319   0.9507   0.3670    0.235   0.2136   0.2313

TRAINING: % variance explained
             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X              98.60    99.18    99.72    99.92    99.97    99.98
nir$protein    48.97    78.17    97.28    98.85    99.06    99.24
```

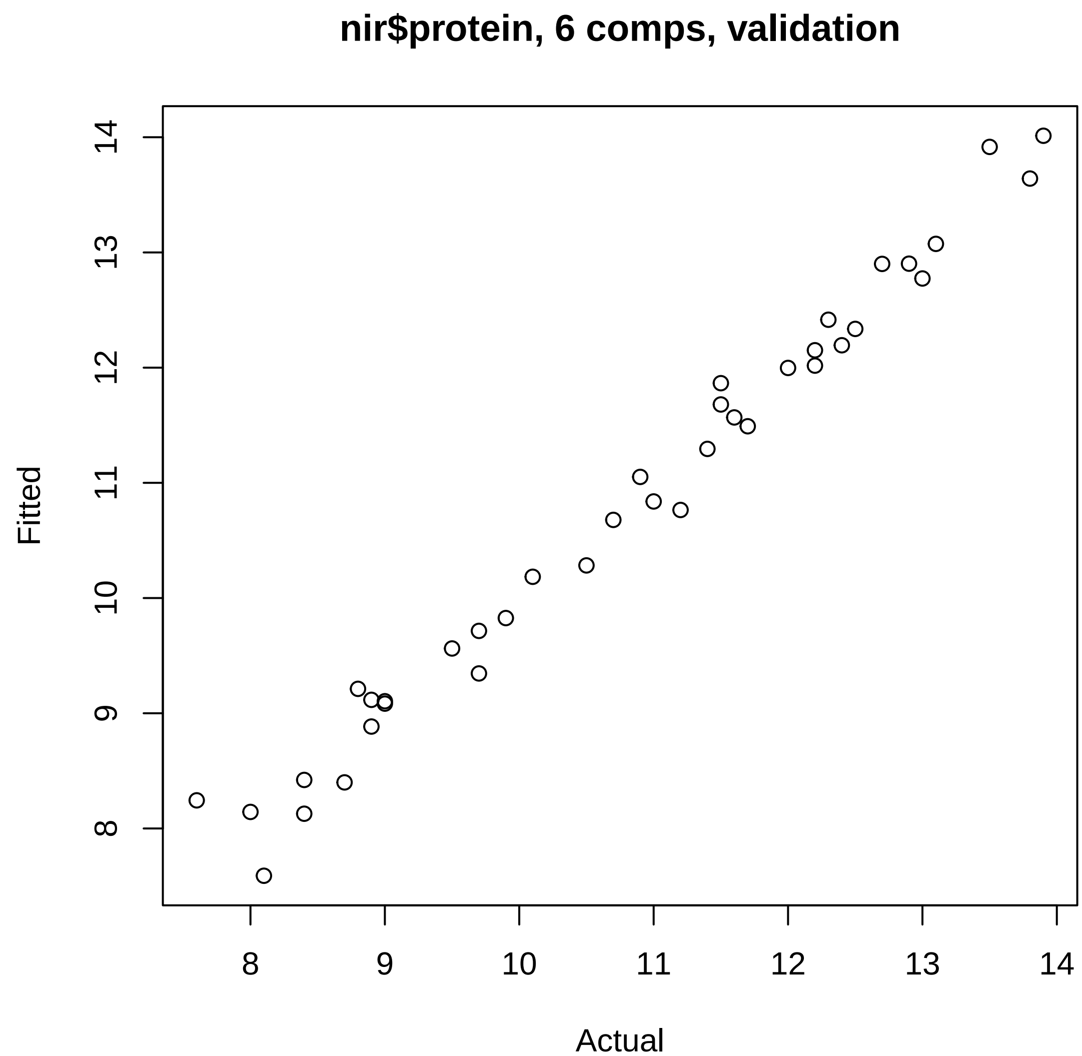
The `ncomp` argument controls up to how many latent PLS factors or components we want to consider. As discussed in previous chapters, cross-validation is convenient here as well to have a more realistic estimate of the goodness of the fit. The cross-validated Root Mean Squared Error of Prediction (RMSEP) suggests that 4 PLS components would be enough. The R^2 for the successive number of PLS components is obtained by

```
# Crude (train) and CV R2
R2(pls, estimate = "all", intercept = FALSE)
```

```
       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
train   0.4897   0.7817   0.9728   0.9885   0.9906   0.9924
CV      0.4540   0.7327   0.9554   0.9823   0.9854   0.9826
```

Goodness of fit using 6 PLS components using a *observed* versus *predicted* plot.

```
# Crude (train) and CV R2
plot(pls, xlab = "Actual", ylab = "Fitted")
```



Thus, we obtain very similar results using PCR or PLS in this case.

The fundamental difference between PCR and PLS is in the way they incorporate the y values into the model. PLS does this directly by involving covariances between y and all the x_i variables. PCR derives its "factors" independently of y, then makes the link by straightforward multiple regression. One consequence of this difference seems to be that to achieve the same level of predictive ability PCR models generally require one, sometimes two, more components than the corresponding PLS model. For both PCR and PLS, the number of terms for the model should be determined by cross-validation. The function `plsr` in the `pls` package does this for us. We have not used cross-validation for the PCR example above simply because it would involve some messy R programming to it by hand. In fact the `pls` package includes a `pcr` function which does do cross-validation for PCR as for PLS, but that function does not incorporate the flexibility of selecting your PCs though. It forces you to use the first k PCs, which as we have seen from our example, is not always the most sensible choice.

In both the PCR and PLS modelling of the NIR data we did not standardise the X values, the NIR wavelengths, because they are all measured on the same scale. It is more common, when the X variables are measured on disparate scales, that we standardise by centring (subtract the mean) and scale (divide by the standard deviation). The argument `scale` does this in the `pls` function. However one must be careful with this operation in high-dimensional data as it would give the same relevance to genuine and noisy signals.

Finally note that, as with PCR, we can convert the PLS model back to the original wavelengths. The R code is similar and the resulting PLS model (not shown) is almost identical to the PCR model.

12.4 Exercises

1. NIR data. Use principal component regression to predict the moisture values. How many components do you need? Show a plot of the predicted versus actual moisture values.
2. Repeat the above exercise using partial least squares.