Data Analysis

Lecture Notes

Univ.-Prof. Dipl.-Ing. Dr.techn.
Peter Filzmoser

Research Unit Computational Statistics

Institute of Statistics and Mathematical Methods in Economics

TU Wien

Vienna, March 2024

Preface

A first step in data analysis should consist of an exploratory approach, in which one tries to identify possible structures in the data in a more informal way. Such an approach is supported by appropriate graphics and diagnostic plots. For example, the histogram is one of the ”oldest” graphical data representations in applied statistics, and it is still considered as a very useful tool for displaying the data. The choice for the best suited graphical methods mainly depends on the data to be analyzed. The amount of data and their dimensionality play an important role in this context.

The dimensionality of the data is often a challenge for the analysis. Although one can analyze every variable separately, information about the relationships between the different variables is usually most interesting and important. While there are many graphical methods for displaying univariate or bivariate data, it is much more tricky to find good tools to graphically analyze higher dimensional data. This is also the point where multivariate statistical methods are indispensible in data analysis, and jointly with appropriate visualizations they form a strong team to gain knowledge out of data.

Our statistical tools build on specific assumptions, such as randomness or independence of the observations, or normal distribution. Thanks to these assumptions many statistical methods are conveniently mathematically tractable, and easy to implement and to use. However, already slight deviations from idealized conditions and assumptions could lead to false and misleading conclusions. Thus, an important goal in data analysis is to find procedures which are resistant to such deviations. Small deviations from the model should only have little impact to the analysis. This course will cover some of these methods, and it also tries to identify the reason why traditional methods could fail.

One of the pioneers of exploratory data analysis was John W. Tukey (1915-2000). He invented various simple yet efficient methods for analyzing statistical data. Many of these methods are still very popular, such as boxplots. In his opinion, a graphical data inspection is crucial for the analysis: *“It is important to understand what you can do before you learn to measure how well you seem to have done it.”* (Tukey, 1977)

The vast amount of data occurring today have heavily influenced the world of statistics. “Big data” is present in many application areas, and the use of traditional statistical methods is often prohibited due to the complexity of the data. On the other hand, approaches such as Deep Learning seem to be more and more successful in real data applications, and at some point one gets the impression that common statistical methods are outdated and of no practical use any more. The reality might be different: Not only the approaches from classical statistics and modern computer science are very diverse, also the data sets and the analysis tasks can be very different. The real challenge is to combine the different concepts for an improved analysis, which could be the typical task within Data Science (which, as the name says, is still science).

The concepts covered in this lecture will be demonstrated at data examples analyzed in the statistical software environment *R*, see *http://www.R-project.org*. Most of the statistical procedures, even the newly developed ones, are implemented in *R*, which is freely available for different computer systems. Most of the graphics in these course notes were generated in *R*. The corresponding *R* commands are printed in `italics` in some of the figure legends.

Relevant literature

J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey (1983). *Graphical Methods for Data Analysis*, Chapman and Hall, New York.

W.S. Cleveland (1987). *The Collected Works of John W. Tukey*, Volume V, Graphics 1965-1985, Chapman and Hall, New York.

W.S. Cleveland (1993). *Visualizing Data*, Hobart Press, Summit, New Jersey.

S.H.C. du Toit, A.G.W. Steyn, and R.H. Stumpf (1986). *Graphical Exploratory Data Analysis*, Springer, New York.

M. Friendly (2000). *Visualizing Categorical Data*, SAS Press, Cary, NC.

J.R. Gessler (1993). *Statistische Graphik*, Birkhäuser, Basel.

D.C. Hoaglin, F.M. Mosteller, and J.W. Tukey (1983). *Understanding Robust and Exploratory Data Analysis*, Wiley, New York.

R. Maronna, D. Martin, and V. Yohai (2006). *Robust Statistics. Theory and Methods*, John Wiley & Sons Canada Ltd., Toronto, ON.

C. Reimann, P. Filzmoser, R.G. Garrett, and R. Dutter (2008). *Statistical Data Analysis Explained. Applied Environmental Statistics with R*, John Wiley & Sons, Chichester.

J.W. Tukey (1977). *Exploratory Data Analysis*, Addison-Wesley, Reading, Massachussets.

K. Varmuza and P. Filzmoser (2009). *Introduction to Multivariate Statistical Analysis in Chemometrics*, CRC Press, Boca Raton, FL.

E.J. Wegman E.J. and D.J. DePriest (1986). *Statistical Image Processing and Graphics*, Marcel Dekker, New York.

Contents

Chapter 1

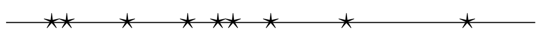
Statistical graphics for univariate data

We assume univariate data. This means that we have n data values (observations) measured at only one variable (=univariate). We denote these data values by $x_1, x_2, \ldots, x_n$.

This chapter will NOT provide an overview about the various possibilities to visualize univariate data, since this would be a lecture on its own. Rather, we aim to discover WHY statistics is involved in visualizing univariate data, and HOW. We will start with very simple things, and also end with very simple things – but they are still very useful.

1.1 Univariate scatterplots

This is probably the most simple option we could think of, to show the univariate data values on one axis. The result could look like this:



There should also be some scale, and a name for the axis. The axis does not necessarily have to be drawn in horizontal direction. This sounds all simple, but try it out in R with the `plot()` command – not so simple.

The points are scattered along the axis, thus the name scatterplot. This display already has difficulties in case of multiple points on one position, e.g. caused by rounding effects, or if there are outliers.

1.1.1 Multiple points

Imagine measurements of body height, given in cm. Usually, these are integer valued, and if n gets bigger, it is more and more likely to obtain identical data values in the sample – these are called multiple points. We don't want to miss this information, thus the scatterplot needs to be modified. There are different options, such as:

Use different symbols, symbol sizes, or symbol colors representing the multiplicity of the points. However, this could quite disturb the visual impression of the data.

Stacking: Multiple points are stacked on top of each other. The result is a barplot-like visualization.

Jittering: For every data value x_i we randomly generate a value y_i in a certain interval, for $i = 1, \ldots, n$, and we represent the pairs (x_i, y_i) jointly in a 2-dimensional plot. The second dimension should appear "small", since the important information is in the first dimension, and a uniform distribution might be appropriate for the random number mechanism.

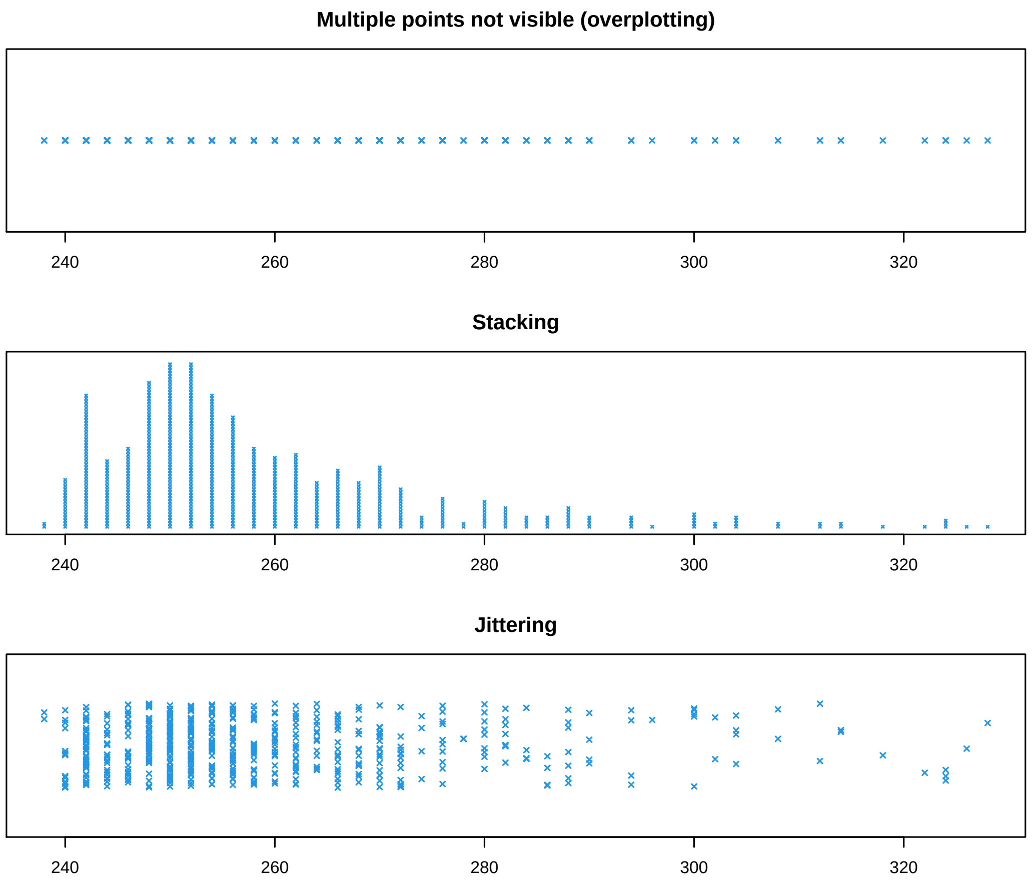


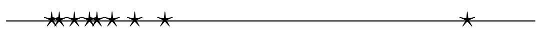
Figure 1.1: Average monthly concentration of ozone in December 2000 at a grid of 24 × 24 measurements in Central America. (`stripchart`)

The plots in Figure 1.1 can be reproduced by:

```
data(ozone,package="plyr")            # load data set
oz72 <- ozone[,,72]                   # select December 2000
stripchart(oz72,method="overplot")
stripchart(oz72,method="stack")
stripchart(oz72,method="jitter")
```

1.1.2 Outliers

Extreme outliers may dominate the scatterplot, and possible structure (groups, gaps, etc.) in the "data majority" might not be visible.



Omitting those outliers can allow to see more details in the data majority. However, we need to make clear somehow that outliers have been omitted, since this is very essential information.

1.2 Histogram

This is a first attempt to "summarize" the univariate data in order to get a better overview about their distribution. Some notation:

$x_1, \dots, x_n$	...	data values (sample)
n	...	number of observations
$k-1$	...	number of histogram bins
$t_1, \dots, t_k$	...	breakpoints for histogram bins
n_i	...	number of observations per interval $[t_i, t_{i+1})$, for $i = 1, \dots, k-1$

The histogram function is defined as

$$H(x) := \sum_{i=1}^{k-1} \frac{1}{t_{i+1} - t_i} \frac{n_i}{n} I_{[t_i, t_{i+1})}(x)$$

with the indicator function

$$I_{[t_i, t_{i+1})}(x) := \begin{cases} 1 & \text{if } x \in [t_i, t_{i+1}) \\ 0 & \text{else.} \end{cases}$$

Here, the histogram function is defined with relative frequencies, but it could also be represented with absolute frequencies. Dividing by the interval length is only important if the interval lengths differ (not equidistant). Usually, we work with equidistant histograms (equal interval lengths), and thus the denominator is irrelevant.

1.2.1 Choice of the interval length

The form of the histogram is heavily determined by the number of histogram bins, thus by the number of intervals, thus by the interval lengths. Note, however, that the form also depends on the starting point t_1 for the first interval, which can be either the minimum of the sample, or any value smaller than the minimum. Equivalently, also the ending point can determine the shape.

Assume that we have equidistant intervals, and thus all interval lengths are equal to $h_n = t_{i+1} - t_i$, for $i = 1, \dots, k-1$. Note that there is an index n involved, because (obviously) the interval length should also account for the number of observations.

There are different "rules" to determine the "optimal" interval length, named after the inventors:

Rule of Sturges: The optimal number k for the optimal interval length $h_n = (t_k - t_1)/k$ is given by

$$k = \lceil \log_2(n) + 1 \rceil,$$

where $\lceil \cdot \rceil$ means rounding up to the next integer. This choice assumes an underlying normal distribution, and although this rule is the default in the *R* function `hist()`, it has been criticized, see `https://robjhyndman.com/papers/sturges.pdf`.

Rule of Scott: This rule is more general, and it assumes an underlying density function f which is continuous, and for which also f' and f'' are continuous and bounded. Then the optimal interval length is given by

$h_n = (\frac{6}{\int_{-\infty}^{\infty} f'(x)^2 dx})^{\frac{1}{3}} n^{-\frac{1}{3}}$.

The idea behind this rule is to minimize the MSE (Mean Squared Error) between the histogram function and the density f,

$\mathrm{MSE}(x) = \mathrm{E}(H(x) - f(x))^2 \qquad \text{for fixed } x$

A error measure for the whole range is the IMSE (Integrated Mean Squared Error):

$\mathrm{IMSE} = \int \mathrm{MSE}(x) dx$,

which is minimized with this rule.
Since f is unknown in general, we usually assume normal distribution, and then this rule simplifies to

$h_n = \frac{3.5s}{\sqrt[3]{n}}$,

where s is the empirical standard deviation.

Rule of Freedman and Diaconis: In case of data outliers, the rule of Scott may lead to a non-optimal choice for the interval length, because s can be inflated by the outliers. The idea is to use a more robust scale estimator, such as the interquartile range IQR, which is the difference between the third and the first quartile, $\mathrm{IQR} = q_{0.75} - q_{0.25}$. The rule is then

$h_n = \frac{2 \cdot \mathrm{IQR}}{\sqrt[3]{n}}$,

which is supposed to be more robust against data outliers.

Figure 1.2 show a comparison of histograms constructed with the rule of Scott or Friedman-Diaconis. Scott's rule is sensitive to outliers.

Figure 1.3 shows a comparison of the optimal interval lengths according to Scott and Friedman-Diaconis for normally distributed data. Note that n plays an important role.

1.3 Density estimation

While with histograms we also try to get an idea about the data distribution by approximating the underlying density function f, we try to estimate this underlying density now more directly. The principle is to look at the local behavior in order to get a smoother approximation.

We consider an interval length h (also called window width). The density $f(x)$ at a location x is estimated based on observations which are in the interval $[x - \frac{h}{2}, x + \frac{h}{2}]$. For the estimation we use a weight function $W(t)$, with the important property

$\int\limits_{-\infty}^{\infty} W(t) = 1.$

The estimated density at a value x is defined as

$$\hat{f}(x) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} W(\frac{x - x_i}{h}).$$

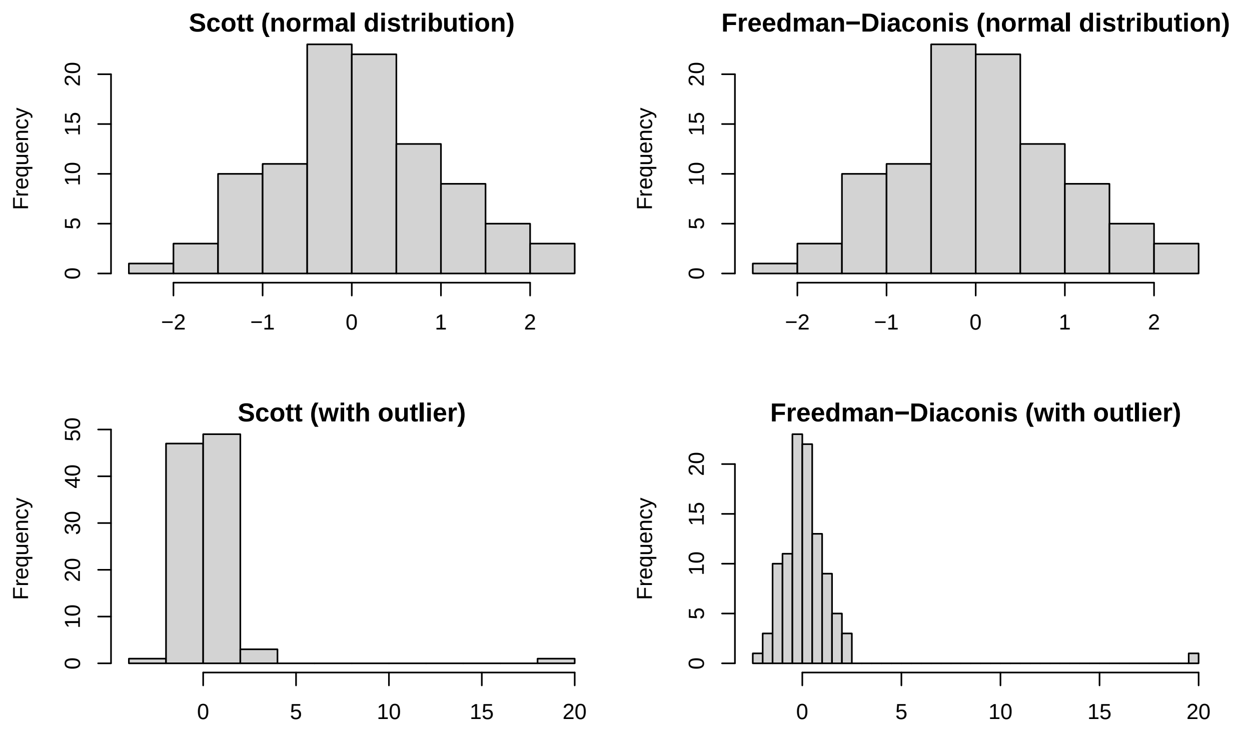


Figure 1.2: Comparison of the rule of Scott and Friedman-Diaconis for normally distributed data without and with outlier.

n	Scott	Freedman-Diaconis
10	1.620	1.252
20	1.286	0.994
30	1.123	0.868
40	1.020	0.789
50	0.947	0.743
75	0.828	0.640
100	0.752	0.582
150	0.657	0.508
200	0.597	0.461
300	0.521	0.403

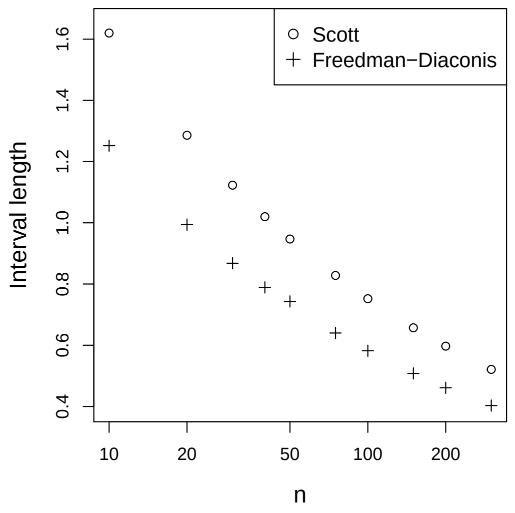


Figure 1.3: Comparison of the optimal interval lengths (in units of the estimated standard deviation) for normally distributed data.

By substituting $t = \frac{x-x_i}{h}$ ($dt = \frac{1}{h}dx$), one can see that

$$\int_{-\infty}^{\infty} \hat{f}(x)dx := \frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{\infty}\frac{1}{h}W(\frac{x-x_i}{h})dx = \frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{\infty}\frac{1}{h}hW(t)dt = 1,$$

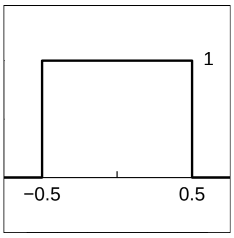
which is what we need (and why the above property was important).

Density estimation is often called *kernel density estimation (kde)*, because the resulting form depends on the choice of the weight function, also called *kernel.* Examples are:

a) Rectangular weight function (boxcar function):

$$W(t) = \begin{cases} 1 & \mid t \mid \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

i.e. $W(\frac{x-x_i}{h}) = 1$ for $|x - x_i| \leq \frac{h}{2}$

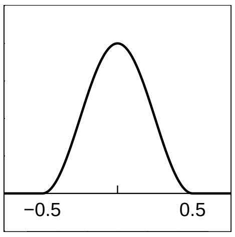


The result is a non-smooth step-function, and the degree of non-smoothness is determined by the number n of observations.

b) Cosine weight function:

$$W(t) = \begin{cases} 1 + \cos 2\pi t & \mid t \mid < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$\int_{-\frac{1}{2}}^{\frac{1}{2}} (1 + \cos 2\pi t)dt = 1 + \frac{1}{2\pi}\sin 2\pi t \mid_{-\frac{1}{2}}^{\frac{1}{2}} =$
$1 + 0 - 0 = 1$



The resulting function $\hat{f}(x)$ is smoother, which might be preferable in practice.

The choice of the window width h is obviously also crucial, and there exist different proposals in the literature which lead to an “optimal” h. We don’t go here into further details.

Figure 1.4 shows the resulting density estimates by using different weight functions.

1.4 Empirical distribution function

Let us go back again to the case of continuous probability densities. We denoted the density function by f. Now we are no longer interested in characterizing this density, but we focus on estimating the distribution function F. We know that there is the relationship

$$F(x) = \int_{-\infty}^{x} f(t)\mathrm{d}t$$

for all x in the corresponding domain.

Consider a sample of n observations, $x_1, \ldots, x_n$. We would like to use this sample in order to estimate F. This can be conveniently done by the empirical distribution function F_n (also called cumulative distribution funtion – cdf), which is defined as

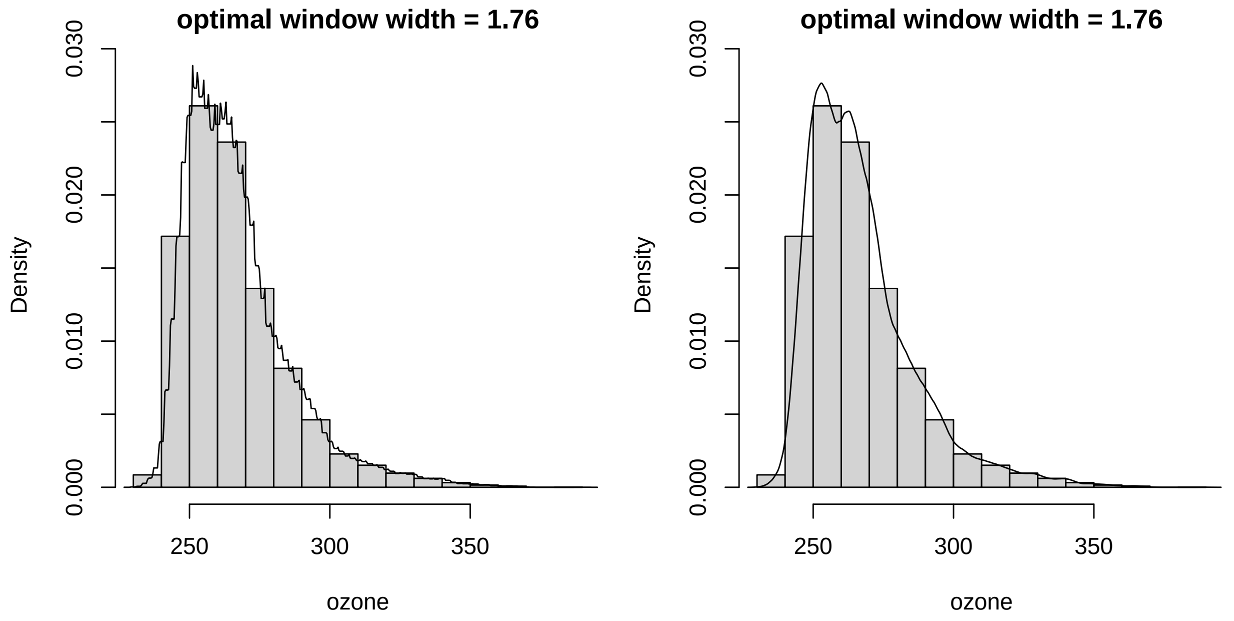


Figure 1.4: Density estimation of the ozone data from Figure 1.1, with boxcar function (left) and cosine function (right). The optimal window width is proposed by the default of the *R* function. (`density`)

$$F_n(x) := \frac{1}{n}\sum_{i=1}^{n} I_{[x_i,\infty)}(x)$$

with the indicator function

$$I_{[x_i,\infty)}(x) = \begin{cases} 1 & \text{if } x \in [x_i,\infty) \\ 0 & \text{else.} \end{cases}$$

In a plot we usually simply show x_i versus $F_n(x_i) = (i - 0.5)/n$, for $i = 1, \ldots, n$, possibly connected by horizontal lines.

Figure 1.5 shows the empirical distribution functions of randomly generated standard normally distributed values. In the left plot we have $n = 30$ values generated, for the right plot $n = 1000$. The dotted line in the plot correspnds to the theoretical distribution function. When n is small, the differences to the theoretical F can be big by accident, whereas for large n the F_n and F are almost indistinguishable. One can show that for $n \longrightarrow \infty$, the **empirical distribution function converges to the theoretical one**.

Figure 1.6 shows data from geochemistry. About 600 soil samples on the Peninsula Kola have been taken from different soil layers, and they have been analyzed for the concentration of various chemical elements. The data sets are available in the R package `StatDA`. In the left plot we see the empirical distribution function of Scandium (Sc) in the soil layer C-horizon. Obviously, for higher concentration values, the laboratory has only reported rounded values, leading to strange artifacts in the plot. The right plot is for Nickel (Ni) in the O-horizon, presented on a log-scale. The Ni concentrations can be very high because of pollution from the industry (there are big smelters in this region), and even in log-scale the distribution looks skewed, and different from a log-normal distribution.

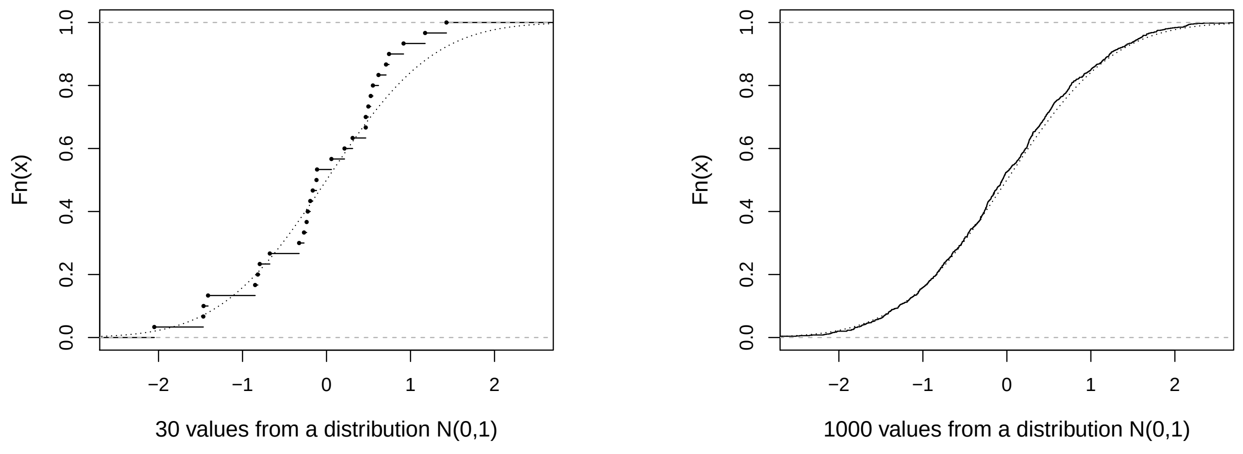


Figure 1.5: Comparison of the empirical distribution function with the theoretical one (dotted curve). (`ecdf`)

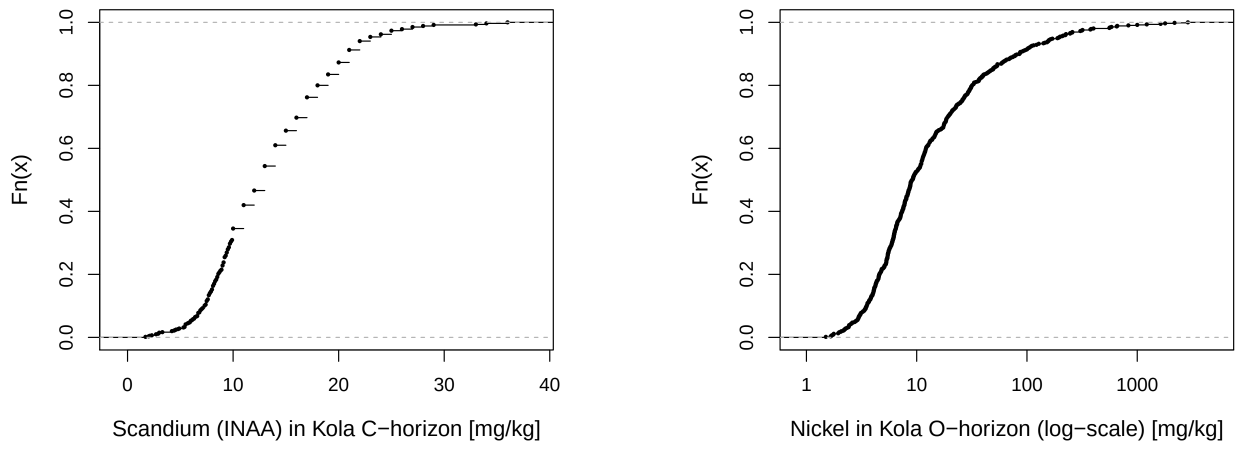


Figure 1.6: Empirical distribution functions. Left: Sc in the C-horizon; right: Ni (log) in the O-horizon.

1.5 Normal probability plot

Although the empirical distribution function is very simple to compute, without the need to estimate any parameter, it is usually hard to see if the underlying distribution has the form of a normal distribution (or any other distribution we would like to compare with). For example, in case of an underlying normal distribution, the points would show a "typical" S-shape. But what is "'typical"? The idea of the *normal probability plot* is to rescale $F_n(x)$, such that the S-shape transforms to a line. Deviations from a line are easier to grasp.

Suppose that the underlying data distribution is F, which is here the distribution function of a normal distribution $N(\mu, \sigma^2)$. Denote the distribution function of the standard normal distribution $N(0,1)$ by Φ, and its inverse by Φ^{-1}. The idea is to plot x against $\Phi^{-1}(F_n(x))$. Since $F_n(x)$ approximates $F(x)$, we have:

$$\Phi^{-1}(F_n(x)) \sim \Phi^{-1}(F(x)) = \Phi^{-1}\left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right) = \frac{x-\mu}{\sigma} = -\frac{\mu}{\sigma} + \frac{1}{\sigma}x,$$

which is the definition of a linear function in x. In other words, the plot shows the points x_i versus $\Phi^{-1}(F_n(x_i))$, for $i = 1, \ldots, n$, and in case of normal distribution, these points should approximately be on a line (depending on n).

One can also estimate the parameters μ and σ of the normal distribution:

Set $x := \mu$. This gives $\Phi^{-1}(F_n(x)) = 0$, and thus $(F_n(x)) = \Phi(0) = 0.5$
Set $x := \mu + \sigma$. This gives $\Phi^{-1}(F_n(x)) = 1$, and thus $(F_n(x)) = \Phi(1) = 0.84$

Thus, at a probability of 50%, cutting the line formed by the points, we read on the other axis an estimate of μ. Similarly, using 84% yields an estimate of $\mu + \sigma$, which identifies the estimate of σ.

As a remark, we can replace the normal distribution in the above derivation by another distribution which we would like to check with, and again the points would have to fall on a line to confirm that the data follow this hypothesized distribution.

Figure 1.7 shows the normal probability plots from the Kola data used in Figure 1.6. The left plot is for Sc, with quite big and systematic deviations from the line, indicating an underlying distribution different from the normal distributions. The rounding effect is again visible, but this time we can see every single data point. The right plot is for Ni, again on a log-scale (but with the original mg/kg values!). Also here we discover systematic deviations from a line, and thus the distribution is even more right-skewed than a log-normal distribution.

1.6 Quantile-quantile plots

For the normal probability plot we used a probability scale, but we could as well just show a scale in terms of quantiles of the standard normal distribution. The shape of the points in the plot would be the same. This is precisely what is done in the quantile-quantile (or simply QQ-) plot.

Similar as for the normal probability plot, the QQ-plot compares an empirical distribution function, computed from the data sample, with a hypothetical one, e.g. with normal distribution. Figure 1.8 shows the idea: A fixed probability p defines the quantiles $q_x(p)$ and $q_y(p)$ for both distribution functions. This is done for several probabilities, and the resulting quantiles are forming the data pairs on the horizontal and vertical axis of the QQ-plot.

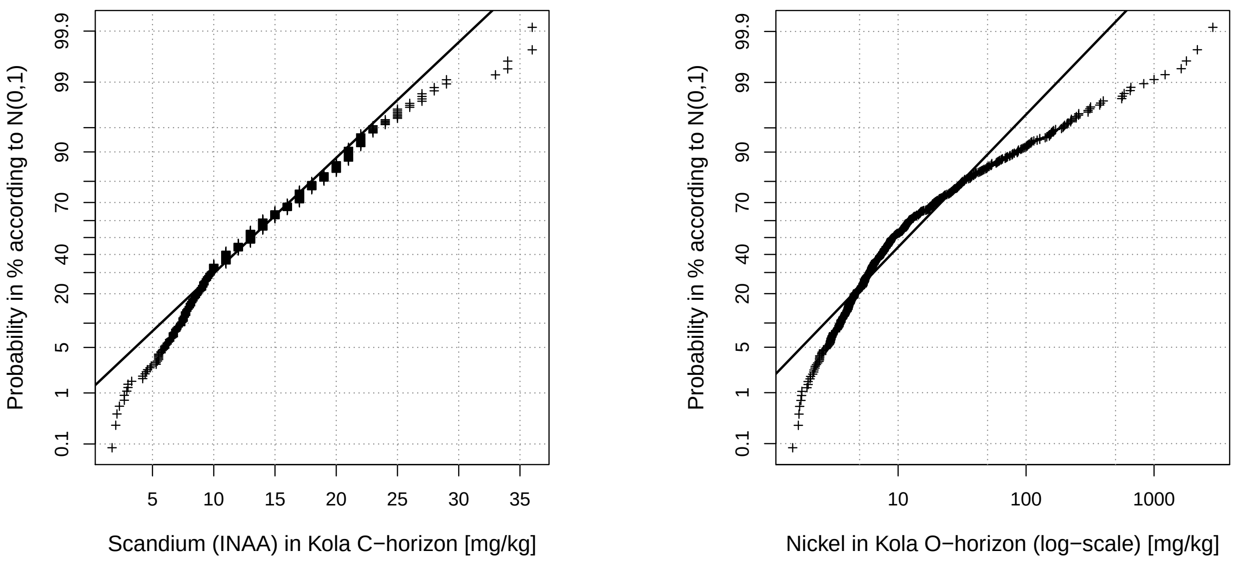


Figure 1.7: Normal probability plot for the Kola data. Left: Sc in C-horizon; right: Ni (log-scale) in O-horizon.

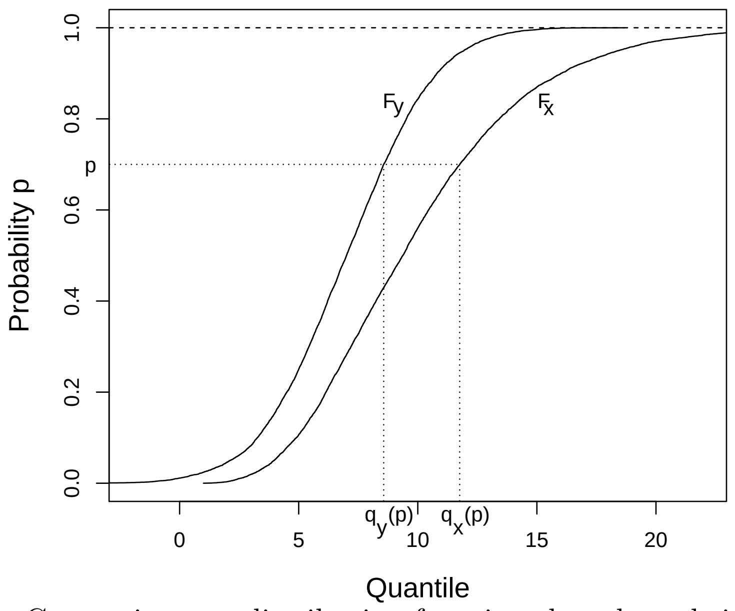


Figure 1.8: Comparing two distribution functions based on their quantiles.

Practically, the probabilities $p_i = \frac{i-0.5}{n}$, for $i = 1, \ldots, n$, are used. If the reference distribution F_x is the standard normal distribution, then the quantiles $q_x(p_i)$ are computed in R by `qnorm((c(1:n)-0.5)/n)`. If F_y refers to the distribution of the data, then the corresponding quantiles $q_y(p_i)$ are simply the sorted data values. The R function `qqnorm()` automatically creates a QQ-plot for a comparison with the normal distribution. `qqline()` fits a line to the points and thus allows to see deviations more clearly.

Figure 1.9 shows a QQ-plot (left) for a data set with annual snowfall accumulations in Buffalo, NY, from 1910 to 1973. Here we have arranged the axes in a different way as for the normal probability plot, because this allows to estimate the parameters is case of normal distribution more conveniently, as outlined below.

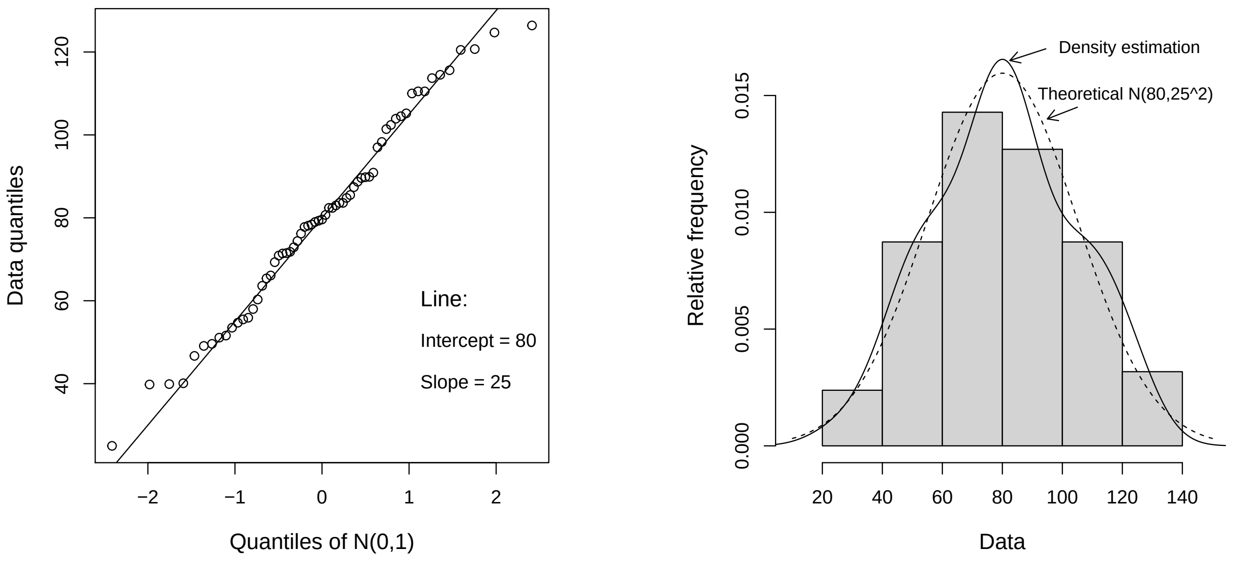


Figure 1.9: Left: QQ-plot of the snowfall data from Buffalo (1910-1973), with the estimated parameters from the normal distribution. Right: Histogram and density estimation for the original data, and density of normal distribution with estimated parameters.

Suppose that F_x is the distribution function of $N(0, 1)$, and F_y is the distribution function of $N(\mu, \sigma^2)$. Then we know that the underlying random variables X and Y have the relationship $Y = \mu + \sigma X$, and for any t we have $F_y(t) = F_x(\frac{t-\mu}{\sigma})$. The quantiles for a probability p are given by $q_x(p) = F_x^{-1}(p)$ and $q_y(p) = F_y^{-1}(p)$.

Now set $t = q_y(p)$:

$$F_y(q_y(p)) = p = F_x\left(\frac{q_y(p)-\mu}{\sigma}\right)$$

$$q_x(p) = F_x^{-1}\left(F_x\left(\frac{q_y(p)-\mu}{\sigma}\right)\right) = \frac{q_y(p)-\mu}{\sigma}$$

$$\Rightarrow \qquad q_y(p) = \mu + \sigma q_x(p)$$

This implies that for distribution families (e.g. normal distribution, rectangular distribution, exponential distribution, ...), which can be distinguished by a location parameter μ and scale parameter σ, the points should fall on a line. The intercept of the line is an estimation for μ, and the slope is an estimation for σ.

This is also illustrated in Figure 1.9, where the left plots shows the line with estimated intercept and slope. The right plots presents the histogram of the data, together with a density estimation. On top of that we can see a theoretical normal density, with the parameters μ and σ estimated from the left plot.

1.6.1 Deviations from the line

Similar to the normal probability plot, we can expect bigger deviations from the line if the sample size is small. Even if the underlying data distribution is from the same distribution family as the hypothetical distribution, the deviations especially in the tails can become bigger if n is small.

However, we are also interested in **systematic deviations**, because this indicates that the data distribution is from a different family, and it could indicate outliers or breaks in the data. Some examples are shown in Figure 1.10.

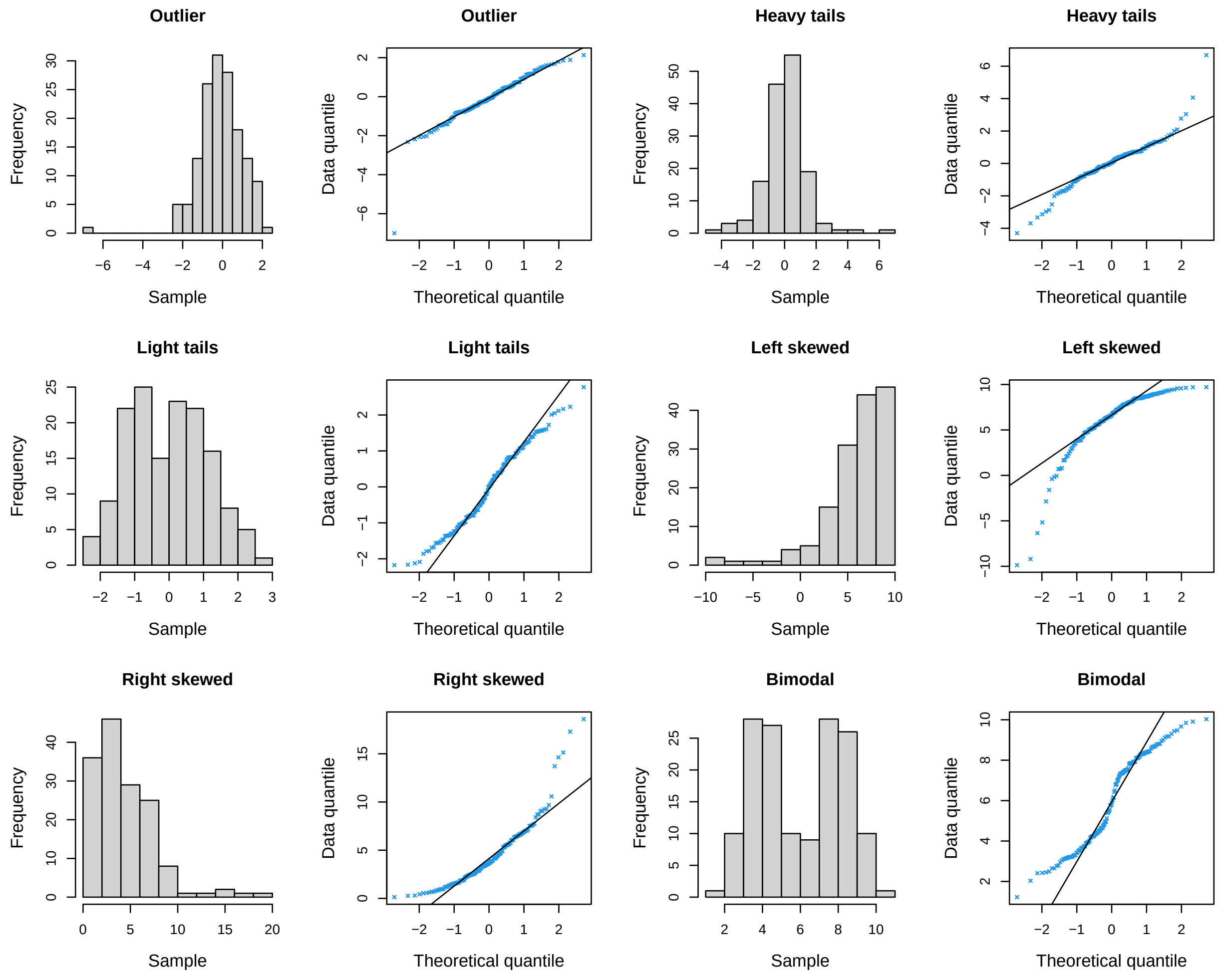


Figure 1.10: Simulated data examples with a distribution that are different from a normal distribution. We can see deviations in the histograms, but particularly systematic deviations in the QQ-plots.

1.6.2 Point-wise confidence envelopes

Even if the data distribution follows the hypothetical one, we can expect a certain variability of the points, especially for small and big quantiles. One can construct point-wise confidence envelopes based on the scale estimation of the empirical quantiles. This can be done by making use of the asymptotic behavior of order statistics, which says that the distribution of the points in a QQ-plot is approximately $N(Q(p), \frac{p(1-p)}{nf(Q(p))^2})$, where $Q(p)$ is the quantile to the probability p, and f is the density of the reference distribution.

Let z_i be the empirical quantile in the QQ-plot. Then an estimation for the standard error s_{z_i} at z_i is given by

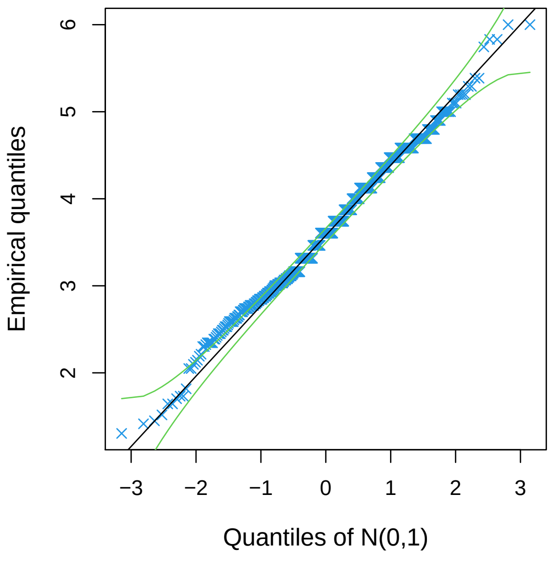
$$s_{z_i} := \frac{\hat{\delta}}{g(q_i)}\sqrt{\frac{p_i(1-p_i)}{n}}$$

with

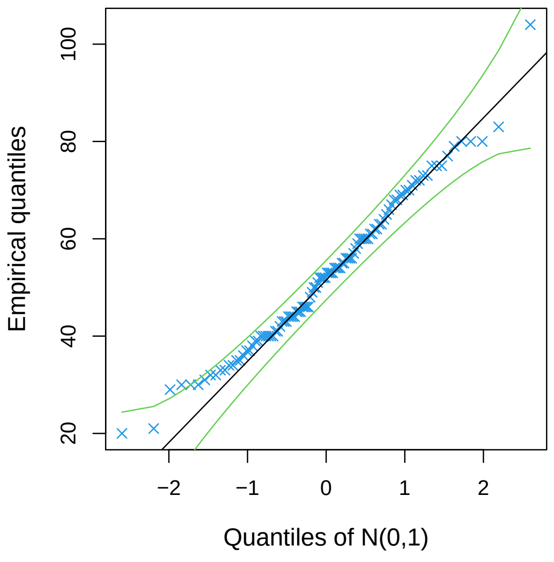
$\hat{\delta}$	estimated slope of the line in the QQ-plot, e.g. $\frac{\hat{q}_{0.75}-\hat{q}_{0.25}}{q_{0.75}-q_{0.25}}$, with the theoretical qantiles $q_{0.75}$ and $q_{0.25}$, and the empirical quantiles $\hat{q}_{0.75}$ and $\hat{q}_{0.25}$.
$g(x)$	density of the reference distribution (e.g. standard normal distribution).
q_i	theoretical quantiles of the QQ-plot.

The point-wise 95% confidence envelope is constructed by adding and substracting point-wise (at the quantiles q_i) the value $2 \cdot s_{z_i}$ to the line.

Figure 1.11 shows QQ-plots with additional confidence envelopes. The left plot is for the Kola C-horizon Sc data. They have even been square-root transformed first in order to better approximate normality. Still, there seems to be a systematic deviation, especially in the lower part of the distribution. The right plot is for the snowfall data from Buffalo, where normality also might not be acceptable. Since accumulated snowfall cannot be negative, the normal distribution might in any case not be an appropriate model distribution.



(a) Sc (sqrt-transformed) in C-horizon



(b) Snowfall data from Buffalo

Figure 1.11: QQ-plots with confidence envelopes. `qqPlot()`, package `car`

1.7 Boxplots

The boxplot has been proposed by the statistician John W. Tukey (1915-2000), and this is an exceptional example of a plot which has been useful in the pre-computer age, and is still used in every-day analyses nowadays. There are slightly different understandings of the definition of how to construct the plot. However, the important guidance is: based on "simple and robust ingredients". This excludes the arithmetic mean and the sample variance. A common definition is graphically shown in Figure 1.12.

We have given a sample of n observations: $x_1, \ldots, x_n$.
The median of the sample is $x_M := median(x_1, \ldots, x_n)$.
$\hat{q}_{0.25}$ and $\hat{q}_{0.75}$ are the sample quantiles 0.25 and 0.75 (1^{st} and 3^{rd} quartile).
IQR $= \hat{q}_{0.75} - \hat{q}_{0.25}$ is the interquartile range, containing the innermost 50% of the data.

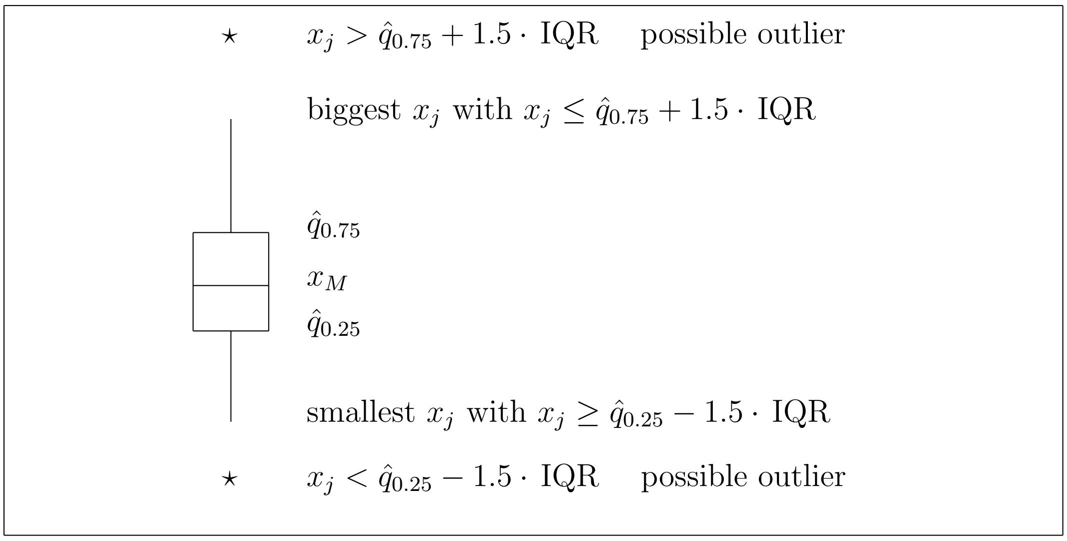

$x_j > \hat{q}_{0.75} + 1.5 \cdot \mathrm{IQR}$ possible outlier
biggest x_j with $x_j \leq \hat{q}_{0.75} + 1.5 \cdot \mathrm{IQR}$
$\hat{q}_{0.75}$
x_M
$\hat{q}_{0.25}$
smallest x_j with $x_j \geq \hat{q}_{0.25} - 1.5 \cdot \mathrm{IQR}$
$x_j < \hat{q}_{0.25} - 1.5 \cdot \mathrm{IQR}$ possible outlier

Figure 1.12: Definition of a boxplot. (`boxplot`)

The "box" is the outer boundary given by the quartiles. The lines which are connected to the box are called "whiskers" (thus the name is often "box- and whiskers-plot"). According to their constructions, boxplots are very informative:

- Estimated center, by the median (robust!).
- Estimated scale, by the IQR (robust!).
- Visual impression of the data skewness, by the position of the median inside the box (robust!).
- Visual impression of the data kurtosis, by the relationship of length of box to length of whiskers (robust!).
- Judgement about possible data outliers (robust!). It can be shown that for normally distributed data, the boxplot outlier rule would (wrongly) "declare" 0.7% of the data as outliers. Note that this rule migth not be very useful for very skewed data, but there is a skew-adjusted version available: `adjbox()`, package `robustbase`.

Boxplots are very useful in combination with previously introduced statistical graphics, as shown in Figure 1.13 for the snowfall data from Buffalo (left) and for the Arsenic concentration data of the Kola O-horizon (right). The latter is in log-scale, and still we can see huge outliers.

Notches for Boxplots: By making use of the theoretical properties of order statistics one can construct a confidence interval (CI) around the median. A 95% CI for the median x_M

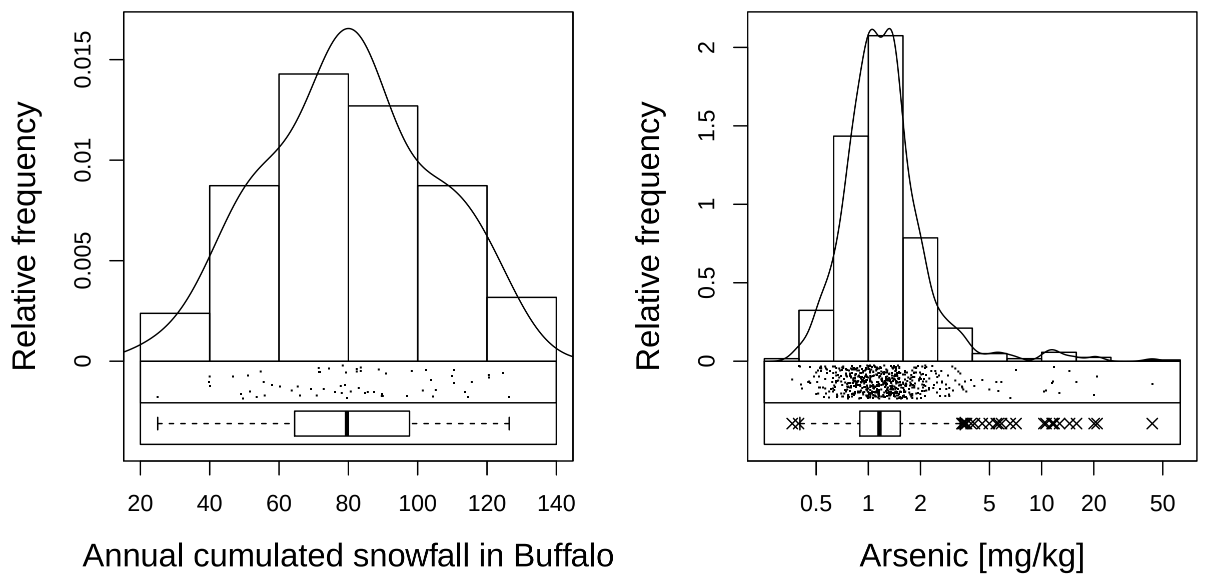


Figure 1.13: Combination of histogram, density estimation, univariate scatterplot, and boxplot, for the snowfall data from Buffalo (left) and the concentration of Arsenic in Kola O-horizon (right). `edaplot()` and `edaplotlog()`, package `StatDA`

is given by:

$$x_M \pm 1.57 \cdot \frac{IQR}{\sqrt{n}}$$

The CI is shown in the boxplot by so-called notches. Note that is cases where n is small, the notches can even be wider than the box, which leads to somehow degenerated forms of the notched boxplots.

Example 1: We consider the data set `crabs` from the *R* package `MASS`. This contains 5 morphological measurements on Leptograpsus Crabs. There are two different species, B for blue, and O for orange, and we have 100 crabs for every species. It would be of interest to distinguish the two species based on the given measurements. Figure 1.14 shows the corresponding boxplots, left plot for the traditional version, right plot with notches. In this context we would use the notch information as follows: If the notches of two boxes (for one measurement) do not overlap, than we would have strong evidence that the medians of the two species differ for this measurement. This seems to be the case for all measurements.

Example 2: The *R* package `ISLR` contains the data set `Carseats`. There is information about 400 shops selling carseats for children (but not only). The variable *Sales* contains the unit sales, and there is further information in *ShelveLoc* about how well the seats are visible in the shop (Bad/Median/Good), *Urban* with information No/Yes if the shop is in a rural or urban region, and *US* with No/Yes whether the shop is based outside or in the US. These categorical variables can be used to groups the sales information and compare the data subsets with boxplots. First results are shown in Figure 1.15, and this can be conveniently computed by using the formula notation in *R*:

```
library(ISLR)
data(Carseats)
attach(Carseats)
```

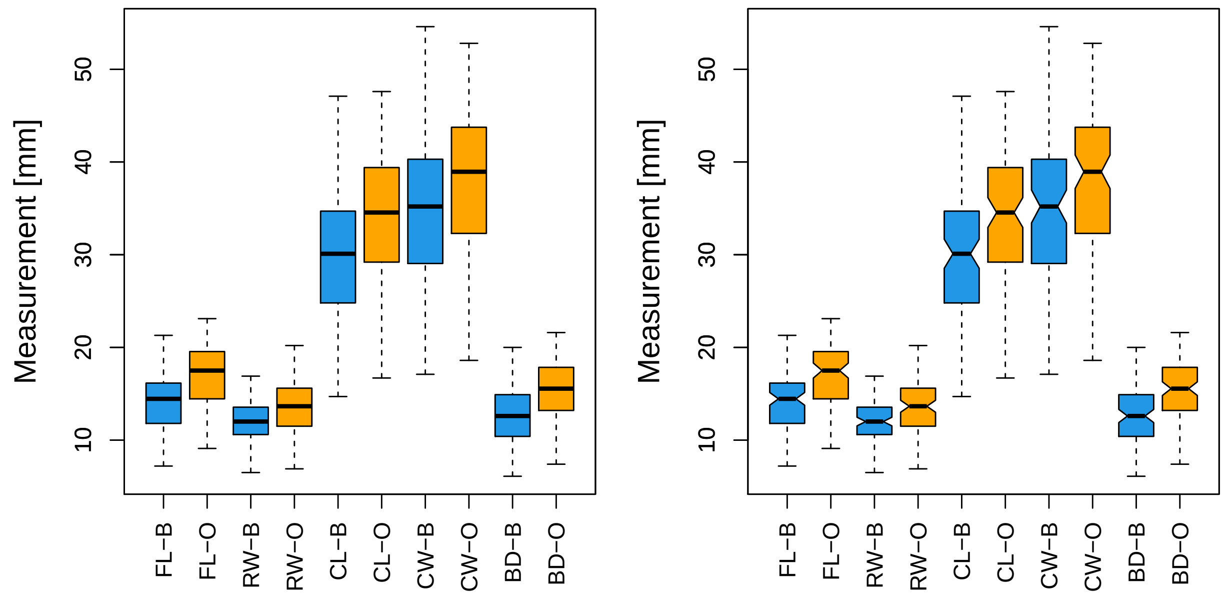


Figure 1.14: Crabs data set with measurements given from two crab species. The right plot is with notches.

```
boxplot(Sales~Urban,notch=TRUE)
boxplot(Sales~US,notch=TRUE)
boxplot(Sales~ShelveLoc,notch=TRUE)
```

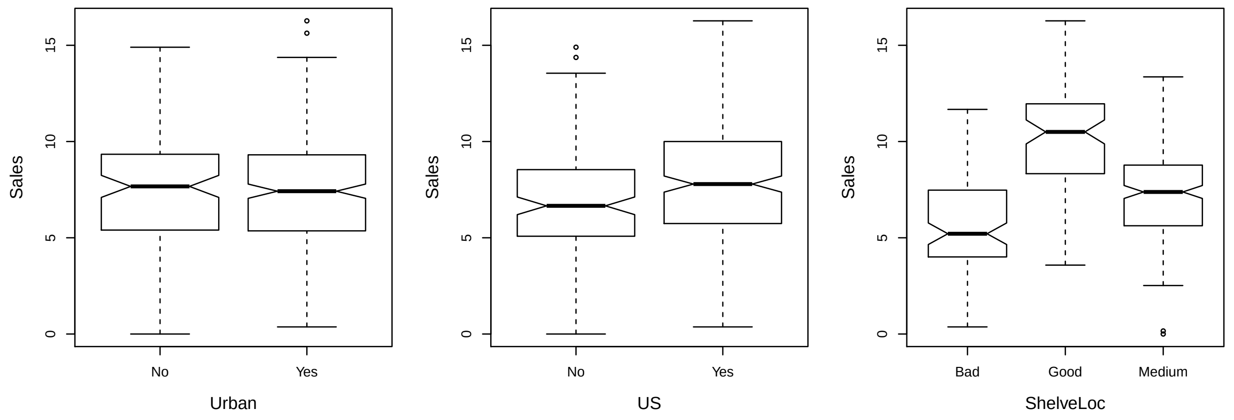


Figure 1.15: Comparison of the sales data for children carseats according to different categories given by additional variables.

Figure 1.15 shows that the median sales unit in rural or urban regions is not significantly different (left), but there are pronounced differences if the location is US or not (middle), and how well visible the seats are in the shop (right).

It is also possible to subset the data according to more than one category, with the code below, and the result is shown in Figure 1.16.

```
boxplot(Sales~US:ShelveLoc,notch=TRUE)
boxplot(Sales~US:Urban:ShelveLoc,notch=TRUE)
```

Figure 1.16 (top) reveals that the best (median) sales units are for shops located in the US, where the carseats are well visible in the shop. The plot below goes even further: in addition

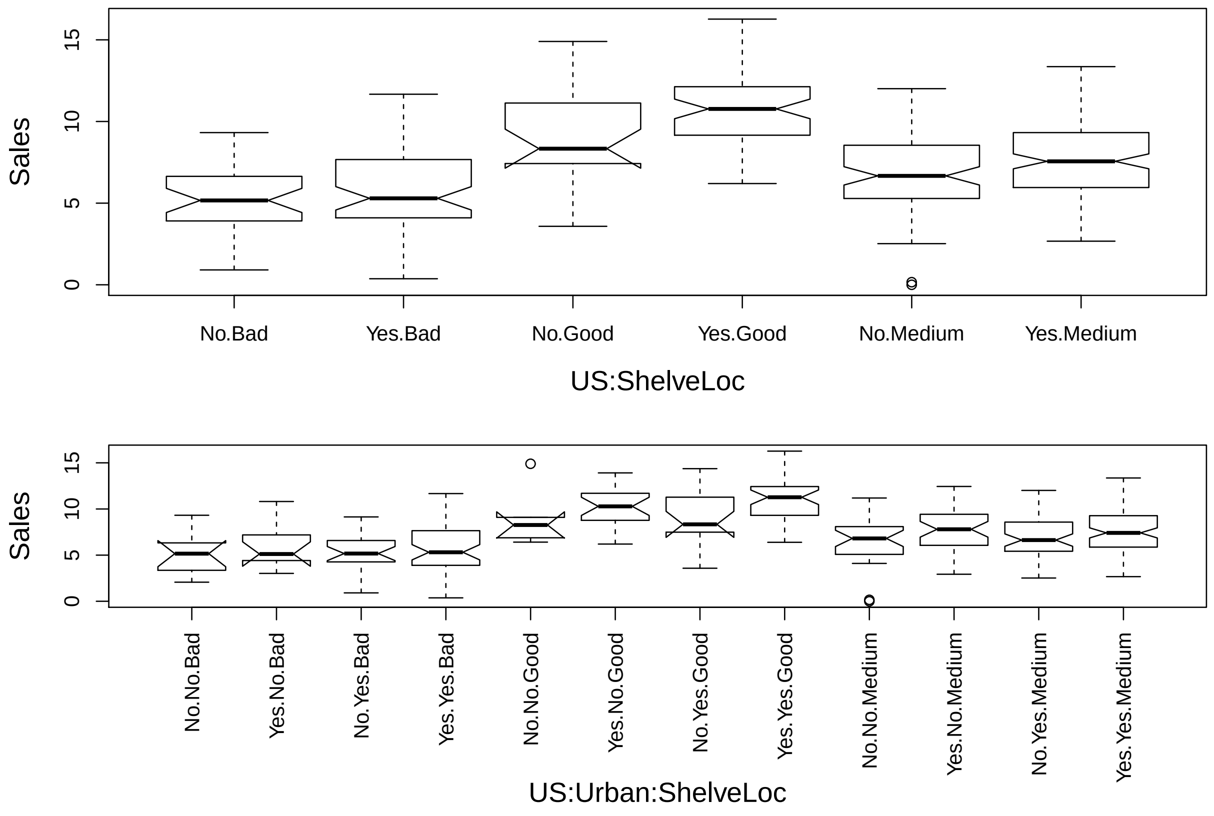


Figure 1.16: Comparison of the sales data for children carseats according to several different categories given by additional variables.

to the findings before we see that shops in urban areas have better sales units than shops in rural regions. Here we also see some "degenerated" boxplots where the notches are wider than the box.

Example 3: The *R* package `laeken` contains the data set `eusilc` with synthetically generated data from real Austrian EU-SILC (European Union Statistics on Income and Living Conditions) data. There is information about the net income of employees, and several additional information on household size, sex, citicenship (Austria, EU, Other), and many more. Figure 1.17 shows so-called violin plots, a modification of boxplots where the shape is determined by a density estimation. This plot compares the net income of data subsets according to sex and citicenship. Not only the medians differ, but also the shapes of the violin plots, and the maxima of the subgroups.

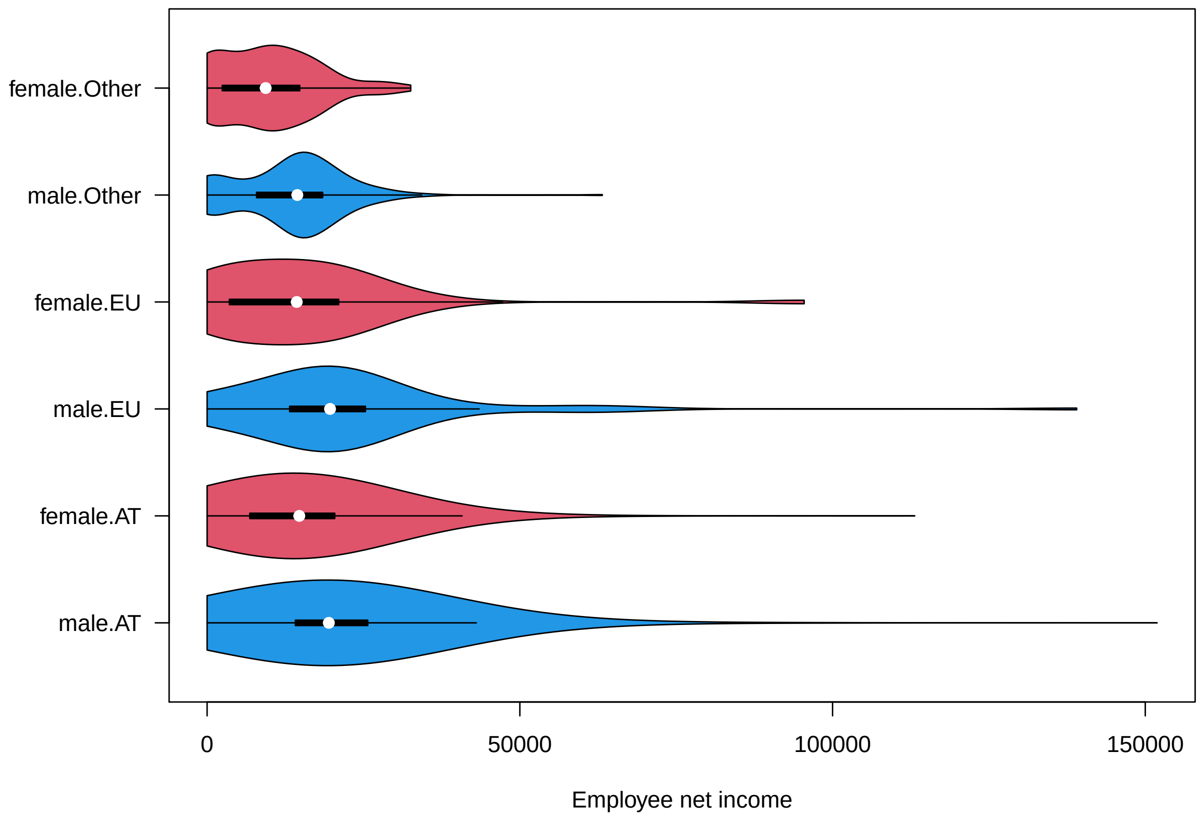


Figure 1.17: Violin plots comparing the net income for females and males, originating from Austria, EU, or outside EU. (`vioplot`, package `vioplot`)

Chapter 2

Statistical estimators for univariate data

The previous chapter mainly focused on statistical graphics in order to get an idea about the data generating process. Often we have been interested in identifying whether or not the data generating process could be based on a normal distribution $N(\mu, \sigma^2)$, which has the location parameter μ and the scale parameter σ. In this chapter we mainly focus on estimating these two parameters by using a data sample $x_1, \ldots, x_n$. It seems quite obvious that if n gets bigger, we should be more precise with our estimation. However, this also depends on other factors, such as data quality issues, or on problems such as outliers in the data.

This means that we shall also be interested in statistical properties of the estimators: Is the estimator really estimating the parameter of interest? How precise is the estimator? How robust is it against contamination?

2.1 "Classical" estimators of location and scale

Let us consider the following problem: We have given a sample $x_1, \ldots, x_n$, and we would like to find a location parameter θ which minimizes the sum of squared residuals

$$\min_\theta \sum_{i=1}^n (x_i - \theta)^2.$$

We can compute the derivative w.r.t. θ, and set the result to zero,

$$-2\sum_{i=1}^n (x_i - \theta) := 0,$$

which identifies the well-known arithmetic mean $\bar{x}$ as our estimator:

$$\hat{\theta} = \bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$$

Indeed, since the second derivative is positive ($= 2$), we obtain the minimum above. So, the arithmetic mean is the least-squares (LS) estimator in the location problem.

For considering theoretical properties we need to switch to the concept of random variables. Let $X_1, \ldots, X_n$ be independent and identically distributed ("i.i.d.") random variables with expectation μ and variance σ^2. Then the arithmetic mean $\bar{X}$ (as a random variable) has also expectation μ, which characterizes an *unbiased estimator*, and thus this estimator is useful to estimate the expectation of a distribution which has generated our sample. The variance of the arithmetic mean is σ^2/n, and thus it gets smaller and smaller with increasing sample size. Small variance is desirable, because higher precision is equivalent to higher confidence in the estimation.

In spite of all these nice properties, the arithmetic mean has also a disadvantage: It is not robust against outliers. In fact, if we would only move a single observation, the estimator will also move in the same direction – in the worst case as far as we want.

The "classical" estimator of scale is the *empirical standard deviation*

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

Its square s^2 is called *empirical variance* or also *sample variance.* Similar as the arithmetic mean, the sample variance is an unbiased estimator for the variance σ^2. Since it is a mean of squared differences to the mean, the sample variance is also sensitive to data outliers. Only one single (extreme) outlier could lead to an arbitrary increase of s.

2.2 "Robust" estimators of location and scale

The "classical" estimators mentioned above may suffer from poor data quality, in particular if outliers are present, or if there are data inconsistencies. But do we have alternatives?

Yes. **Robust estimators of location** are for example:

- **Trimmed mean:** Denote by $x_{(1)}, \ldots, x_{(n)}$ the sample sorted in ascending order, thus $x_{(1)} \le x_{(2)} \le \ldots \le x_{(n)}$. the α-trimmed mean is defined for the parameter $0 \le \alpha < 0.5$ as

  $$\bar{x}_\alpha = \frac{1}{n-2g}\left(x_{(g+1)} + \ldots + x_{(n-g)}\right),$$

  where $g = \lfloor n\alpha \rfloor$, and $\lfloor k \rfloor$ means to take the integer part of k.

  Thus, we take the average without considering the smallest and largest α-proportion of data values. Note that these values are not "thrown away", since their information is still needed for sorting. Typical values of α are 10% or 20%, depending on data quality.

- **Median:** This is the "innermost" value of the sorted data, and it can be computed as

  $$x_M = median(x_1, \ldots, x_n) = \begin{cases} x_{(\frac{n+1}{2})} & \text{if } n \text{ is odd} \\ \frac{x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}}{2} & \text{if } n \text{ is even} \end{cases}$$

  Note that the arithmetic mean was identified as the LS estimator in the location problem. The median is the so-called L_1 estimator, minimizing

  $$\min_\theta \sum_{i=1}^{n} |x_i - \theta|.$$

Since both estimators involve fewer observations (directly), we lose precision. One can show that under normality, the variance of x_M is about 50% higher than that of $\bar{x}$. In other words, in order to attain the same precision, we would need about 1/3 more data for x_M.

On the other hand, both estimators achieve robustness against outliers: For the median we would have to replace 50% of the data values (on one end of the distribution) in order to make this estimator useless. For the α-trimmed mean these would be α% of the values. This also leads to a "quantification" of robustness:
The **breakdown point** of an estimator is given as the smallest fraction of observations which can be replaced by any arbitrary values, such that the estimator gives a useless result.
The breakdown points for our location estimators are:

arithmetic mean $\bar{x}$:	0%
α-trimmed mean $\bar{x}_\alpha$:	α%
median x_M:	50%

It is clear that the breakdown point in this setting cannot be higher than 50%, because we always want to focus on the data majority for the estimation.

And yes. **Robust estimators of scale** are for example:

- **Interquartile range:** This is simply the range of the "innermost" 50% of the data

$$\text{IQR} = \hat{q}_{0.75} - \hat{q}_{0.25},$$

where $\hat{q}_{0.75}$ is the 3^{rd} and $\hat{q}_{0.25}$ the 1^{st} quartile, estimated from the sample.

Note that IQR is not a *consistent* estimator for the parameter σ. Roughly speaking, for a consistent estimator we would like to have (with probability 1) that the estimated parameter comes very close to the true one, even if the sample size gets arbitrarily large. For example, for the standard normal distribution, where $\sigma = 1$, the IQR gives a value of 1.35 (for large n). Thus, in order to make this estimator consistent for σ for normal distribution, we have to divide by 1.35, leading to our robust scale estimator

$$s_{\text{IQR}} = \frac{\text{IQR}}{1.35},$$

which has a breakdown point of 25%.

- **Trimmed standard deviation:** Similar to the α-trimmed mean $\bar{x}_\alpha$ we can define the α-trimmed standard deviation with $0 \le \alpha < 0.5$ and $g = \lfloor n\alpha \rfloor$ as

$$s_\alpha = c_\alpha \cdot \sqrt{\frac{1}{n-g-1}\left(\tilde{x}^2_{(1)} + \ldots + \tilde{x}^2_{(n-g)}\right)},$$

with the values $\tilde{x}_i^2 = (x_i - \bar{x}_\alpha)^2$, which are sorted in ascending order, thus $\tilde{x}^2_{(1)} \le \tilde{x}^2_{(2)} \le \ldots \le \tilde{x}^2_{(n)}$. Note that here we just have to trim the biggest deviations. The constant c_α makes the estimator consistent under normality, but it depends on α. For example, for $\alpha = 0.1$ we have $c_{0.1} = 1.52$.

- **Median Absolute Deviation (MAD):** This scale estimator is defined as

$$\text{MAD} = \underset{1 \le i \le n}{\textit{median}} \mid x_i - x_M \mid,$$

thus as the median of the absolute deviations from the median, leading to the maximum breakdown point of 50%. The estimator is consistent under normality only after correction:

$$s_{\text{MAD}} = \frac{\text{MAD}}{0.675} = 1.483 \cdot \text{MAD}$$

A drawback of s_{MAD} is its low efficiency: Under normality, its variance is roughly 2.7 times higher than that of the empirical standard deviation.

- Q_n **estimator:** This scale estimator also attains maximum breakdown point of 50%, but also high efficiency (the variance is only higher by a factor of 1.2 compared to the classical estimator).

  The Q_n is essentially defined as the first quartile of the absolute pairwise differences,

  $$Q_n = \{|x_i - x_j|; i < j\}_{(k)}$$

  where (k) is the k-th value of the sorted absolute pairwise differences (in ascending order), with $k = \binom{h}{2} \approx \binom{n}{2}/4$ and $h = \lfloor n/2 \rfloor + 1$.

  Q_n is a consistent estimator for σ under normality only with a correction factor:

  $$s_{Qn} = 2.219 \cdot Q_n$$

Example: Here a simple example how to compute the estimators:

Sample:	2.1	3.7	2.6	5.8	1.6	1.1	32.7	4.7	3.1	4.8
Ordered sample:	1.1	1.6	2.1	2.6	3.1	3.7	4.7	4.8	5.8	32.7

$\bar{x} = 6.22$

For $\alpha = 0.1$: $g = \lfloor 10 \cdot 0.1 \rfloor = 1 \Rightarrow \bar{x}_{0.1} = \frac{1}{10-2}(x_{(2)} + \ldots + x_{(9)}) = \frac{1}{8}28.4 = 3.55$

$x_M = \frac{(3.1+3.7)}{2} = 3.4$

$s = \sqrt{\frac{1}{10-1}\sum_{i=1}^{10}(x_i - \bar{x})^2} = 9.4$

$\hat{q}_{0.25} = 2.1$

$\hat{q}_{0.75} = 4.8$

$s_{\text{IQR}} = \frac{4.8-2.1}{1.35} = 2.0$

$s_{0.1}$:

$\tilde{x}_i^2 = (x_i - \bar{x}_{0.1})^2$: 6.00, 3.80, 2.10, 0.90, 0.2025, 0.0225, 1.32, 1.56, 5.06, 849.7

$\tilde{x}_{(i)}^2$: 0.0225, 0.2025, 0.90, 1.32, 1.56, 2.10, 3.80, 5.06, 6.00, 849.7

$s_{0.1} = 1.52\sqrt{\frac{1}{9-1}\sum_{i=1}^{9}\tilde{x}_{(i)}^2} = 2.46$

MAD:

$x_i - x_M$:	-2.3	-1.8	-1.3	-0.8	-0.3	0.3	1.3	1.4	2.4	29.3
$\mid x_i - x_M \mid_{(i)}$:	0.3	0.3	0.8	1.3	1.3	1.4	1.8	2.3	2.4	29.3

$\text{MAD} = \frac{1.3+1.4}{2} = 1.35$

$s_{\text{MAD}} = \frac{\text{MAD}}{0.675} = 2.0$

Qn:
absolute differences $|2.1-3.7|, |2.1-2.6|, \ldots, |2.1-4.8|, |3.7-2.6|, \ldots, |3.1-4.8|$ sorted; take the k-th value, with $k = \binom{6}{2} = 15$.
The 15th biggest value is $Q_n = 1.5$, and thus $s_{Qn} = 3.33$.
The R function `Qn()` from the package `robustbase` gives a value of 2.397 because of a correction factor for small samples.

2.3 Univariate outlier identification

The term "outlier" has been mentioned already several times, but so far no "definition" has been provided. What is an outlier?

From a statistical point of view we would have a data generating process in mind, which generates our "regular" data, but the outliers have been generated from a different process.

Figure 2.1 illustrates this idea: The 30 regular data in blue are generated from the "blue" distribution, in this case a distribution N(10,2). The 5 outliers in pink are generated from the "pink" distribution N(15,1). What we observe in practice are 35 data points, and we have to find an outlier cutoff value which separates the regular data from the outliers. If we set this cutoff at 15, we would identify only one outlier. Setting it to 14 would find 4 out of 5 outliers, but also declare a regular observation as an outlier. So, the outlier cutoff value will usually be always a compromise between correctly identifying the outliers (TP = true positives), and incorrectly declaring regular observations as outliers (FP = false positives). Since outliers could also appear in the lower part, we should also have a cutoff value on the left-hand side.

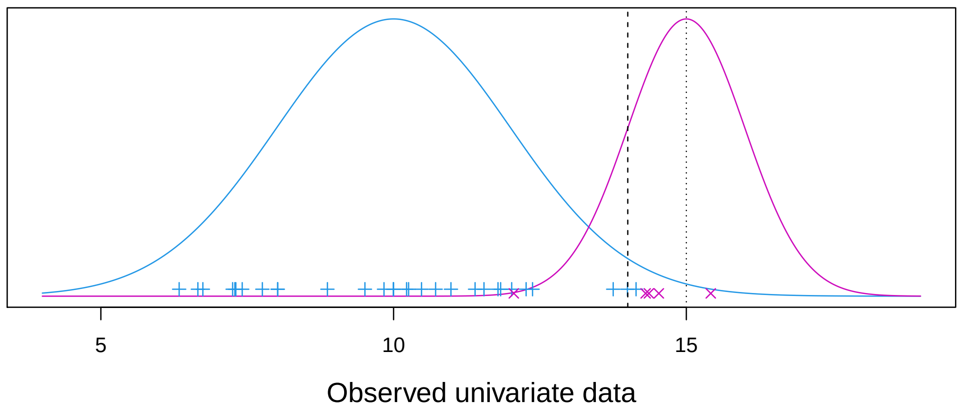


Figure 2.1: Possible outlier generating process for univariate data.

When observing univariate data it would usually not be feasible to estimate the locations and scales of the two (or more) data generating distributions, as indicated in Figure 2.1. Rather, we try to estimate the parameters for the distribution which generated the "data majority", and then try to come up with an appropriate outlier cutoff.

Outlier identification rules:

- **Boxplot rule:** According to the definition of a boxplot, see Figure 1.12, outliers would be identified as observations which are

  smaller than $\hat{q}_{0.25} - 1.5 \cdot \text{IQR}$ or

  bigger than $\hat{q}_{0.75} + 1.5 \cdot \text{IQR}$.

  Note that this rule is also robust against outliers!

- **Rule $\hat{\mu} \pm 2 \cdot \hat{\sigma}$:** We know that the "inner 95%" of the normal distribution N(μ,σ^2) are given by the interval $[\mu - 1.96 \cdot \sigma, \mu + 1.96 \cdot \sigma]$. Values outside this interval could be potential outliers. Thus, we just need to estimate location μ and scale σ in order to identify the outliers:

  $\bar{x} \pm 2 \cdot s$ ... not appropriate, since both estimators are sensitive to outliers.

$x_M \pm 2 \cdot s_{\text{MAD}}$ ... appropriate, since both estimators are robust against outliers.

Of course, any other robust estimators of location and scale could be considered as alternatives.

Simulation study: Below we present a small simulation study in order to evaluate the rules mentioned above.

- **Clean data:** The data are generated from a single normal distribution, thus without any outliers (in the above sense). However, the sample size is varied from 10 to 10.000 observations, which has an effect on the estimated parameters. The results are presented in Figure 2.2 left. From theory we know that the probability to be outside of $\mu \pm 2 \cdot \sigma$ is 4.6%, and this is what we see for big sample size for the "classical" rule $\bar{x} \pm 2 \cdot s$ as well as for a robust version thereof. However, for small sample size this can look quite different. Moreover, we know from theory that the boxplot rule incorrectly identifies 0.7% of the observations as outliers, but this also changes if the sample size is smaller.
- **Varying the amount of contamination:** Here, the data are generated as indicated in Figure 2.1, with a normal distribution generating the regular data, and a normal distribution with different mean and variance generating the outliers. The proportion of the generated outliers is varied, which is shown on the horizontal axis of Figure 2.2 right. The vertical axis shows the percentage of outliers identified. In these simulations, the sample size is "big". The "classical" rule seems to be quite useful for an outlier proportion of up to 15%, but this depends very much on how the outliers are simulated. The robust version is extremely stable, since it involves very robust estimators of location and scale. The boxplot rule is robust against 25% of contamination, as it can be expected from theory.

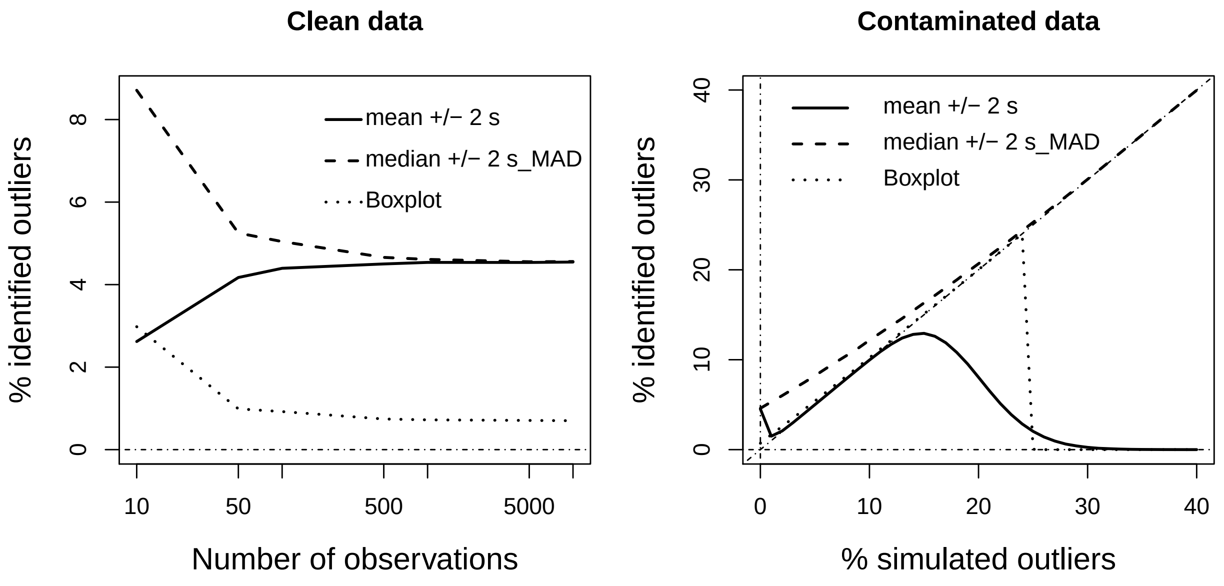


Figure 2.2: Results from simulation studies with outlier detection rules. Left: effect of sample size for clean data; right: effect of amount of contamination.

Chapter 3

Some extensions to bivariate data

Bivariate data consist of observations which have been measured simultaneously on two variables. If we call the variables x and y, then we have pairs (x_i, y_i) for $i = 1, \ldots, n$, where n is the number of observations.

3.1 (Bivariate) Scatterplots

This is much more natural than univariate scatterplots, since we can show the pairs of values directly in a coordinate system, with a horizontal and a vertical axis.

Figure 3.1 left shows an example of a scatterplot, and the right plot also shows the marginal distributions of the variables with histograms and density functions. These bivariate data have been simulated from a bivariate normal distribution, which implies that the marginal distributions of x and y are also normal distributions. However, in contrast to univariate data analysis we also obtain important information about the (bivariate) relationship between x and y. The scatterplot could also show more complex relationships, structures, patterns, etc., which are to be explored. For data sets with even more variables we could look at scatterplots of all pairs of variables (scatterplot matrix) to make the variable associations visible.

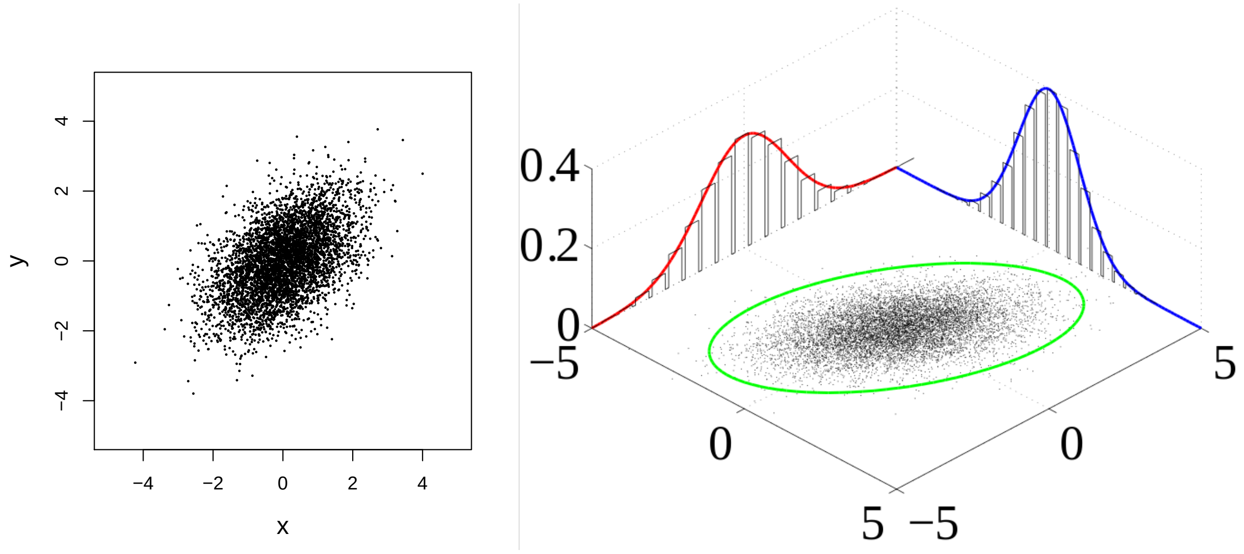


Figure 3.1: Bivariate scatterplot for normally distributed data.

3.2 Multiple points in scatterplots

Similar to the univariate case we can have identical mutliple points also in the bivariate case. There are various options to cope with this situation, and some are listed here:

- Random noise added to the data (*jittering*),

  $$\tilde{x}_i := x_i + \theta_x u_i$$
  $$\tilde{y}_i := y_i + \theta_y v_i,$$

  where u_i, v_i are generated independently from a uniform distribution in (-1,1), and θ_x, θ_y are fixed, e.g. $\theta_x = 0.02(x_{max} - x_{min})$ and $\theta_y = 0.02(y_{max} - y_{min})$.
  The scatterplot shows the pairs $(\tilde{x}_i, \tilde{y}_i)$, for $i = 1, \ldots, n$.
  Jittering (for one variable) can be done by the function `jitter()` in *R*.

- Sunflower plot:
  Multiple points are represented by specific symbols:
  1 value • 2 values | 3 values ⅄ 4 values + 5 values ⋇ etc.
  A sunflower plot can be done in *R* with the function `sunflowerplot()`.
  Of course, one could think of any other version where plot symbols are modified according to the multiplicity of the points (increasing size, different color, etc.).

- Show the original data, and add as "background" a bivariate density estimation (see next section) which color-codes the multiplicity of the observations.

Example: Table 3.1 shows a data set of managers employed at Bell Laboratories: reported is their age in the year 1982, and the number of years since they finished their studies.

Table 3.1: Data of managers at *Bell Labs*: Age in the year 1982 (A) and number of years (Y) since finishing studies.

A	Y	A	Y	A	Y	A	Y	A	Y	A	Y	A	Y	A	Y
35	12	42	16	44	18	47	21	50	28	54	28	57	32	59	31
36	10	42	19	44	19	47	21	51	22	54	29	57	37	59	32
36	12	43	15	44	19	47	21	51	27	54	29	58	23	59	33
36	14	43	17	44	20	47	21	52	19	54	30	58	27	59	34
37	10	43	17	45	13	47	23	52	25	55	25	58	28	59	35
37	12	43	17	45	15	47	25	52	25	55	27	58	31	60	27
38	10	43	17	45	20	47	26	52	26	55	29	58	32	60	28
38	14	43	20	45	20	48	18	53	22	55	29	58	33	60	33
39	10	43	21	45	21	48	21	53	23	55	30	58	33	61	34
39	14	44	9	45	21	48	23	53	24	55	31	58	33	61	40
39	15	44	12	46	18	48	26	53	27	55	31	58	34	62	43
40	12	44	14	46	18	49	20	53	30	55	33	58	34	62	35
40	14	44	16	46	19	49	22	53	31	55	33	59	25	63	30
41	10	44	16	46	20	50	17	53	32	56	27	59	28	63	41
41	17	44	17	46	21	50	21	54	21	56	28	59	29	64	40
42	8	44	17	47	18	50	23	54	27	56	28	59	30	64	41
42	12	44	18	47	21	50	24	54	28	56	30	59	30	66	43

The data set contains multiple points mainly because of rounding artifacts.

Figure 3.2 shows the data which essentially reveal a linear relationship between both variables. Since there are still more details to be seen, we are using several options which allow more focus on two point clusters:

- upper left plot: original data
- upper right plot: jittered data
- lower left plot: sunflower plot
- lower right plot: original data superimposed with density estimation (see next section for details).

It might be a matter of "taste" which option is preferred, but the two clusters with higher point-density seem to indicate very active periods for hiring managers.

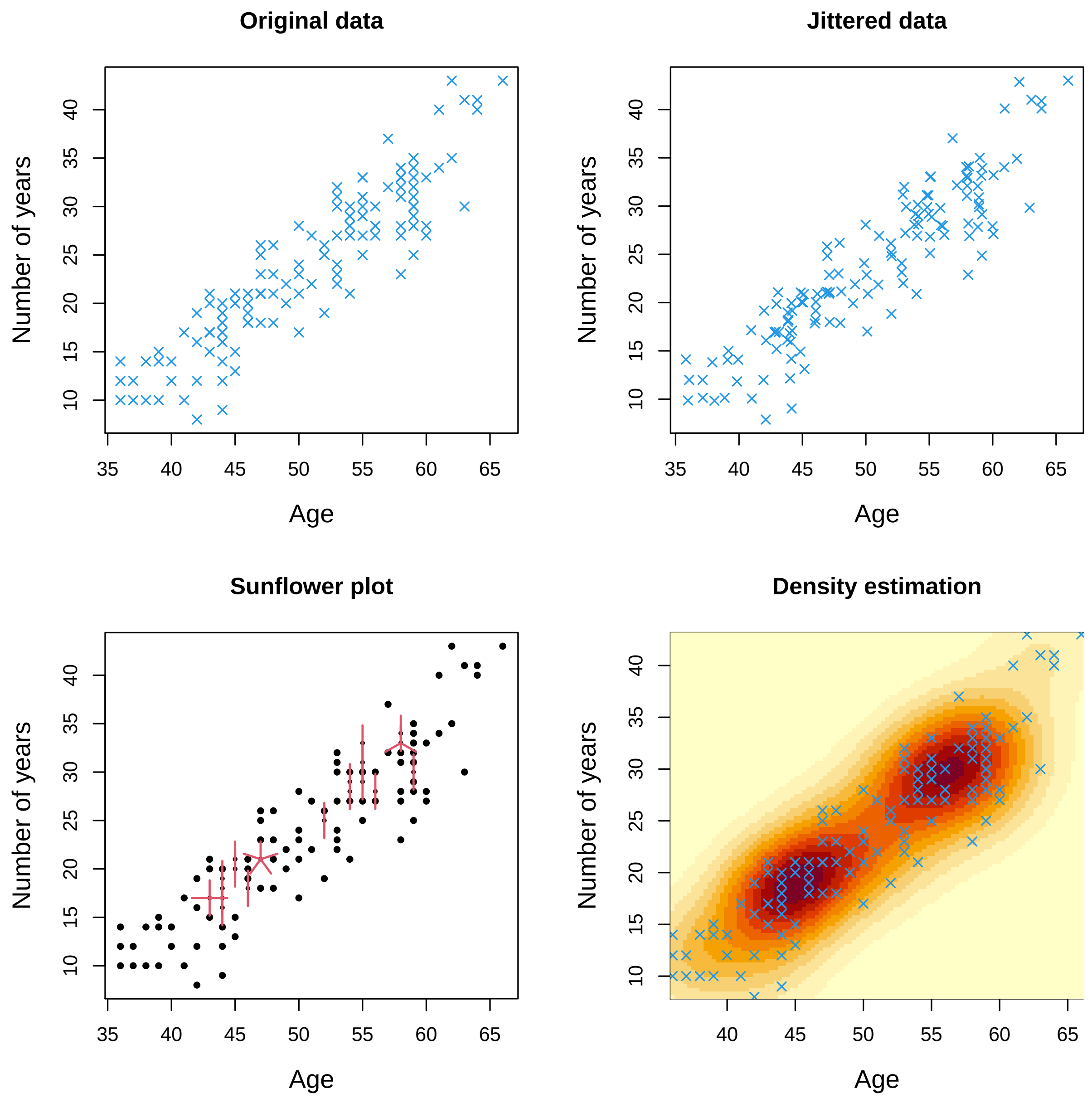


Figure 3.2: Scatterplot of the Bell Labs data set with different options how to deal with multiple points.

3.3 Bivariate density estimation

Similar to the univariate case it is also possible in the 2-dimensional case to estimate the density function. The basic principle is the same: select some local region – this time in two dimensions – such as a square or a circle, and evaluate a given weight function W (kernel).

Based on a given window width h (which could be different in the two coordinates x and y), we obtain the bivariate density estimation as:

$$\hat{f}(x, y) := \frac{1}{h^2 n} \sum_{i=1}^{n} W\left(\frac{x - x_i}{h}, \frac{y - y_i}{h}\right)$$

Again, the weight function needs to be chosen such the the integral is 1, in order to guarantee that the integral (in both direction) of the density estimation is 1. Examples are:

a) *Boxcar* weight function:

$$W(u, v) = \begin{cases} \frac{1}{\pi} & u^2 + v^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

b) Cosine weight function:

$$W(u, v) = \begin{cases} \frac{1+\cos(\pi\sqrt{u^2+v^2})}{\pi} & u^2 + v^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

This choice will lead to as smoother density estimation.

In practice, the density estimation is obtained at a grid defined by the intersection of a number of horizontal and vertical points. In *R* this can be done by the function

- `kde2d()` of the package `MASS`. The function returns the grid points along the two axes, and a matrix with dimensions given by the number of grid points, with the density estimations.
- This output can be further processed with the function `image()`, which plots a smoothed interpolated image, where the density information can be color-coded.
- The output can also be processed with the function `contour()`, which computes isolines by interpolating the density information at defined levels.
- A further plot option is with `persp()` for perspective plot, which shows the density estimation in the third dimension. The viewing perspective needs to be defined by two parameters.
- A plot in 3D can also be obtained by using the package `rgl`. This package allows to interactively rotate the plot. Functions for visualization are `plot3d()` or `surface3d()`.

Example: The *R* package `MASS` contains the data set `geyser`, with data from the "Old Faithful" geyser in the Yellowstone National Park. Two variables have been observed in the period August 1-15, 1985: the eruption time (in minutes), and the waiting time for this eruption (in minutes).

The density estimation has been done with a bivariate normal weight function, and the number of grid points along both axes has been set to 50. Figure 3.3 shows the original data points (upper left), the density estimation shown as image plot (upper right), and presented as contour plot (lower left). The lower right plot combines all plot versions.

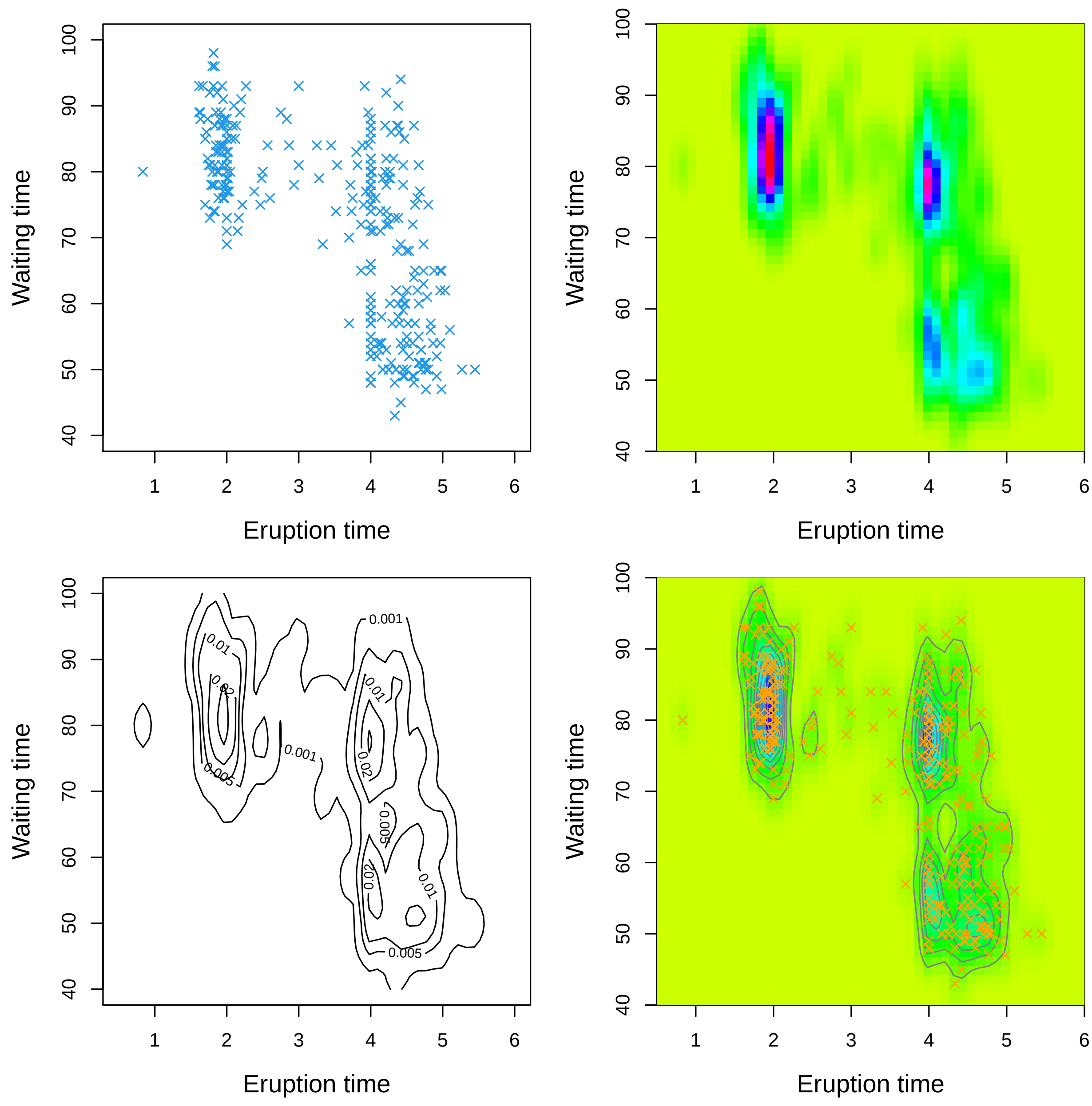


Figure 3.3: Density estimation with the `geyser` data: original data (top left), image plot of the density estimation (top right), contour plot (bottom left), combined plots (bottom right).

Figure 3.4 presents two options to visualize the 3D structure as a result of the function `kde2d()`. Left a perspective plot, right a plot using the package `rgl`, where an interactively rotated version has been exported to a static graph.

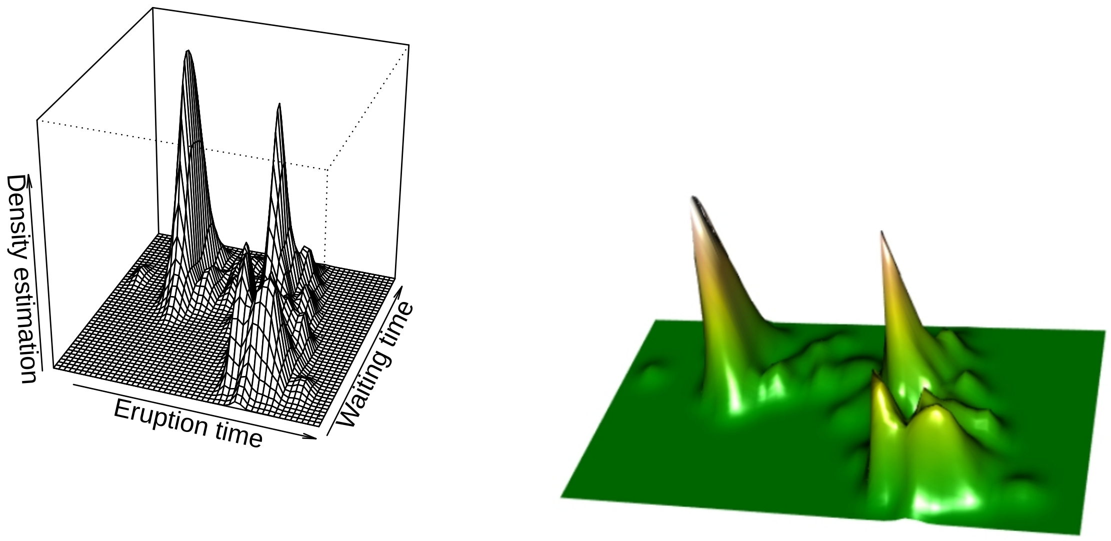


Figure 3.4: Three-dimensional presentation of the density estimation for the `geyser` data.

Chapter 4

Estimation of linear trends

The name of this chapter could also be **linear regression**, or, since we focus on problems where outliers could be present, **robust linear regression**.

We first consider the "simple" regression case, where we have one input varable x, often called "independent" variable, or "explanatory" variable, and an output variable y, also called "dependent" variable, or "response" variable. The goal is to estimate a linear function which allows to predict y based on given x information. This linear function $f(x)$ can be defined with two parameters, an intercept α and a slope β, which gives

$$f(x) = \alpha + \beta x.$$

Now we can use the function to model our response:

$$y = f(x) + \varepsilon = \alpha + \beta x + \varepsilon$$

However, there is an additive error term ε, because we cannot expect in general that our response is related to the explanatory variable precisely in a linear manner – there will always be some randomness involved, which implies random deviations from an exact linear relationship.

Let us consider now a sample of size n, thus pairs of values for both variables, $(x_1, y_1), \ldots, (x_n, y_n)$. The sample should be used to estimate the parameters of the liner function. We call the estimated parameters $\hat{\alpha}$ and $\hat{\beta}$, and the estimation should result in a good approximation,

$$y_i \approx \hat{\alpha} + \hat{\beta} x_i \qquad i = 1, \ldots, n.$$

With the estimated parameters we also obtain the fitted values of the response y as

$$\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i \qquad i = 1, \ldots, n.$$

The fitted values are now exactly on a line, given by the estimated parameters. The discrepancies between fitted and observed response are called **residuals**, defined as

$$r_i = y_i - \hat{y}_i = y_i - \hat{\alpha} - \hat{\beta} x_i \qquad i = 1, \ldots, n.$$

As mentioned before, the residuals should be small (in general). However, what does this mean? Every single residual should be small? The sum of the residuals? A function of the residuals? This needs to be defined more precisely, and this will determine also the statistical properties of the estimated regression parameters.

4.1 Least-squares (LS) estimator

This is the most widely used option which is also taught in school and in many courses at university. As the name suggests (but not very clearly!), the objective function to be minimized is the sum of the squared residuals. Those parameters α and β which yield the smallest value of the objective function are called LS estimators $\hat{\alpha}_{LS}$ and $\hat{\beta}_{LS}$. This can be written as:

$$(\hat{\alpha}_{LS}, \hat{\beta}_{LS}) = \text{argmin}_{\alpha,\beta} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 = \text{argmin}_{\alpha,\beta} \sum_{i=1}^{n} r_i^2$$

This minimization problem has a unique solution for the estimated parameters, and the solution is even given by explicit formulas:

$$\hat{\beta}_{LS} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \quad \text{and} \quad \hat{\alpha}_{LS} = \bar{y} - \hat{\beta}\bar{x}$$

So far this had not much to do with statistics, since we could simply view this task as an optimization problem. However, from a statistical point of view we are also interested in statistical properties of the estimated parameters. For this we would have to switch to a notation where the error term is considered as random variable, resulting also in the response as random variables (howevver, we keep the same letters here):

$$y_i = \alpha + \beta x_i + \varepsilon_i \qquad i = 1, \ldots, n$$

An important assumption is that ε_i are i.i.d. random variables, following the distribution $\mathrm{N}(0, \sigma^2)$. Thus, we have the *residual variance* σ^2, which is the same for all indexes i. This is also an important parameter to be estimated, because it is used to construct confidence intervals and statistical tests for $\hat{\alpha}_{LS}$ and $\hat{\beta}_{LS}$, as well as prediction intervals.

If the mentioned assumptions are valid, the LS estimator is unbiased, and among all unbiased estimators it has the smallest possible variance.

If the mentioned assumptions are violated, the confidence intervals and statistical tests for the LS estimator can be misleading. Even more, if there are severe outliers, this estimator could be far from indicating the linear relationship between x and y for the data majority. This is illustrated with artificial data in Figure 4.1:

- **Vertical outliers:** These are outliers only in y direction, but not in the input variable(s). The upper left plot shows outlier-free data, together with the LS line. The top right plot contains an additional vertical outlier, and again the LS line to the modified data. We can see that the line has been "attracted" by the outlier.
- **Leverage points:** The bottom left plot shows the same data as the top left. The bottom right plot contains a "bad" leverage point, which is a value that is outlying in the input variabe x, but also deviates heavily from the linear trend. This point has a strong effect on the LS estimator. A "good" leverage point would be an outlier in x which follows the linear trend. Such an outlier could even stabilize the LS fit.

From a robustness point of view we would like to fit a model to the data majority, but we would not necessarily like to accommodate every single observation. This, however, is the problem of LS regression. The LS criterion involves every single observation (with the same weight) in terms of the squared residuals, and the square makes outliers even more dominating. There could be several options how to "design" a robust estimator: different weights, non-squared residuals, etc., and some of those ideas will be discussed in the following sections.

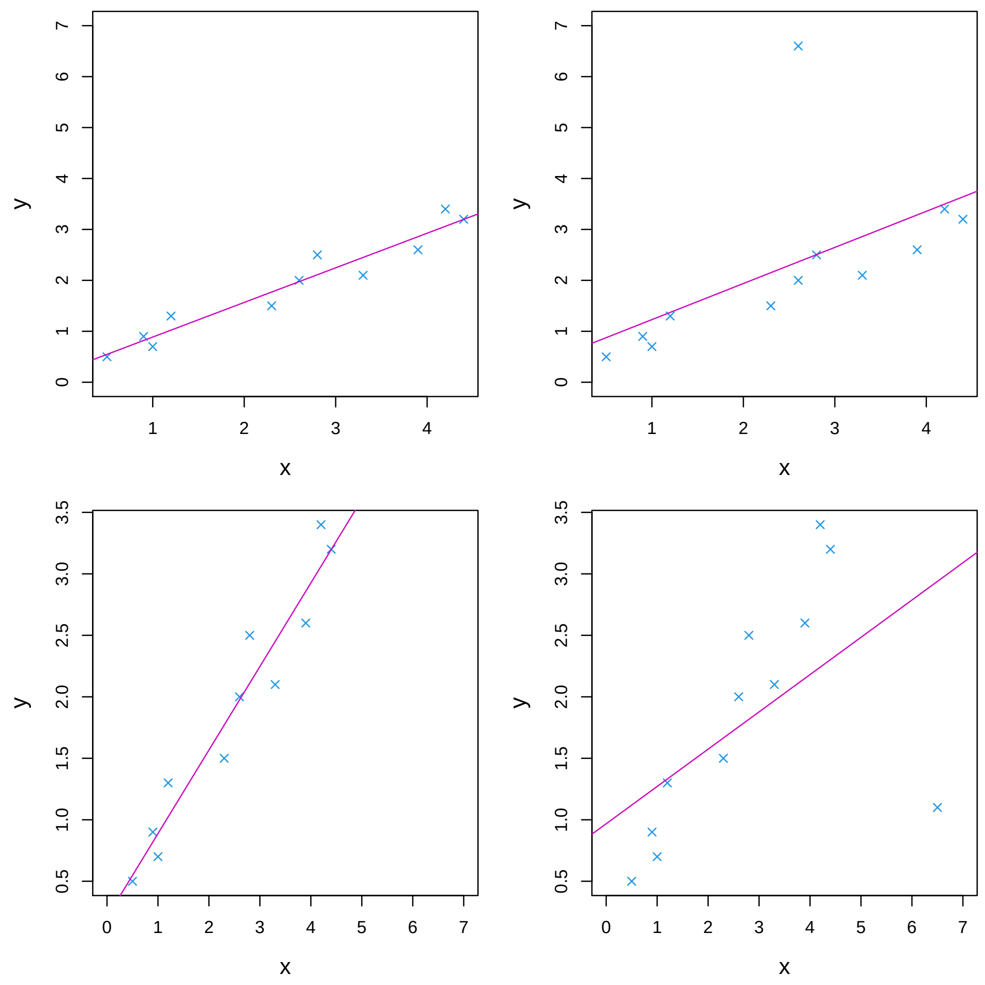


Figure 4.1: Artificial data with LS fit: top right plot contains a vertical outlier, bottom right plot contains a bad leverage point. (`lm`)

4.2 Robust regression line after Tukey

This is a very simple method, developed by John Tukey (1970), which estimated the the parameters of the regression line iteratively:

1. Sort the data pairs (x_i, y_i) according to their x-values.
2. Group the data pairs into 3 groups of about equal size:

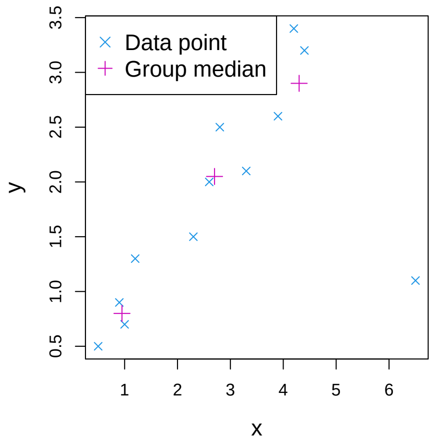
Group L (left)	pairs (x_i, y_i) with n_L smallest x-values
Group M (middle)	pairs (x_i, y_i) with n_M intermediate x-values
Group R (right)	pairs (x_i, y_i) with n_R biggest x-values

   where $n_L + n_M + n_R = n$.

   Pairs with identical x-values should be assigned to the same group.
3. Compute medians of the x- and y-values in the single groups:
   (x_L, y_L), (x_M, y_M), (x_R, y_R) with

   $$x_L = \underset{(x_i,y_i)\epsilon L}{median}\ x_i$$
   $$y_L = \underset{(x_i,y_i)\epsilon L}{median}\ y_i$$

   etc.



4. $\hat{\beta}_0 := \frac{y_R - y_L}{x_R - x_L}$ first estimate of β

   $y = \alpha + \beta(x - x_M) \quad \Rightarrow \quad \alpha = y - \beta(x - x_M)$

   $$\begin{aligned}\hat{\alpha}_0^{(*)} &= \frac{1}{3}[(y_L - \hat{\beta}_0(x_L - x_M)) + y_M + (y_R - \hat{\beta}_0(x_R - x_M))] \\ &= \left(\frac{1}{3}(y_L + y_M + y_R) - \hat{\beta}_0\frac{1}{3}(x_L + x_M + x_R)\right) + \hat{\beta}_0 x_M := \hat{\alpha}_0 + \hat{\beta}_0 x_M\end{aligned}$$

5. Residuals
   $r_i^{(0)} := y_i - (\hat{\alpha}_0^{(*)} + \hat{\beta}_0(x_i - x_M))$ for $i = 1, \ldots, n$
6. Steps 3-5 with data pairs $(x_i, r_i^{(0)})$ for $i = 1, \ldots, n$ yields line
   $r_i^{(0)} = \hat{\alpha}_1 + \hat{\beta}_1(x_i - x_M)$ and residuals $r_i^{(1)} := r_i^{(0)} - (\hat{\alpha}_1 + \hat{\beta}_1(x_i - x_M))$
7. Iteration: continue as in step 6, i.e.
   $r_i^{(j)} \rightarrow \hat{\alpha}_{j+1}, \hat{\beta}_{j+1} \rightarrow r_i^{(j+1)}$ until the estimated parameters are below a tolerance value.
8. $\hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 + \hat{\beta}_2 + \cdots$
   $\hat{\alpha} = \hat{\alpha}_0 + \hat{\alpha}_1 + \hat{\alpha}_2 + \cdots$

Figure 4.2 shows the resulting lines after every iteration. After the thirs iteration the line does not change any more. Since the estimation is based on medians in 3 data groups, the breakdown point of the Tukey regression estimator is 1/6.

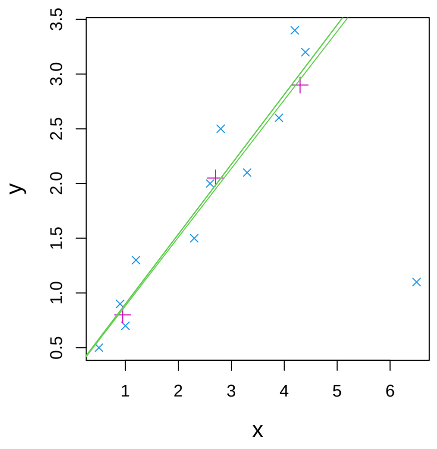


Figure 4.2: Robust Tukey regression line after 3 iterations. (`line`)

4.3 Robust regression line after Theil

The idea of Theil (1950) was to consider the median of the slopes given by all pairs of data points (all x values need to be distinct).

$$\hat{\beta}_{ij} := \frac{y_j - y_i}{x_j - x_i} \qquad 1 \le i < j \le n \qquad \text{i.e. } \binom{n}{2} = \frac{n(n-1)}{2} \quad \text{slopes.}$$

The Theil estimator for the regression parameters is then given by:

$$\hat{\beta}_T := \underset{1 \le i < j \le n}{\mathit{median}} \; \hat{\beta}_{ij} \qquad \hat{\alpha}_T := \underset{1 \le i \le n}{\mathit{median}} \; (y_i - \hat{\beta}_T x_i)$$

The breakdown point of the Theil estimator can be derived as follows. We need to identify the smallest number k of observations that is sufficient to cause breakdown of the estimator:

$\binom{k}{2}$	...	number of outlying slopes caused by outliers
$k(n-k)$	...	number of outlying slopes linking outliers with regular data
$\binom{n}{2}/2$	...	number of slopes to cause breakdown of the median

$$\begin{aligned}
\binom{k}{2} + k(n-k) &= \frac{\binom{n}{2}}{2} \\
\frac{k(k-1)}{2} + k(n-k) &= \frac{n(n-1)}{4} \\
k^2 - k + 2nk - 2k^2 &= \frac{n(n-1)}{2} \\
-k^2 + 2nk - k &= \frac{n^2(1-\frac{1}{n})}{2} \\
\left(\frac{k}{n}\right)^2 - 2\frac{k}{n} + \frac{k}{n^2} &= -\frac{1}{2} + \frac{1}{2n} \\
\left(\frac{k}{n}\right)^2 - 2\frac{k}{n} + \frac{1}{2} &\approx 0 \qquad \text{for large } n \\
\frac{k}{n} &= 1 \pm \sqrt{1 - \frac{1}{2}} \\
\frac{k}{n} &= 1 - 0.71 \quad = \quad 0.29
\end{aligned}$$

4.4 Repeated median regression

The repeated median regression estimator of Siegel (1982) is also based on medians of pairwise slopes, but they are computed repeatedly, by fixing one index at a time:

$$\hat{\beta}_{RM} := \underset{1 \le i \le n}{median} \; (\underset{\substack{1 \le j \le n \\ j \neq i}}{median} \; \hat{\beta}_{ij}) \qquad \hat{\alpha}_{RM} := \underset{1 \le i \le n}{median} \; (y_i - \hat{\beta}_{RM} x_i)$$

This estimator was the first one to achieve a breakdown point of 0.5. This is easy to see:

> The outer median would lead to a non-sense result if at least $n/2$ values from the inner median are non-sense. There are two cases:
> a) the inner median is computed with all pairwise slopes to a "good" observation: we would need at least $n/2$ slopes to produce a non-sense result.
> b) the inner median is computed with all pairwise slopes to an outlying observation: this can produce a non-sense result.
> Overall we see that at least $n/2$ values need to be outliers in order to cause breakdown.

Remark: This concept can be extended to the case where we have more input variables $x_1, \ldots, x_p$ for the regression task (see last section in this chapter). However, the computational burden gets high for large n. Moreover, when rescaling the variables, the regression estimator does not lead to the same rescaled result.

4.5 Least Median of Squares (LMS) regression

The LMS regression estimator has been introduced by Rousseeuw (1984), and it results from robustifying the LS criterion:

$$(\hat{\alpha}_{LMS}, \hat{\beta}_{LMS}) = \text{argmin}_{\alpha,\beta} \, \text{median}_i r_i^2 \; .$$

Thus, the sum in LS regression is replaced by the median, which leads to a breakdown point of 50%.

This estimator also works for multiple input variables, and the estimator also approriately rescales when rescaling our data.

Algorithm: The solution can be approximated by a so-called subsampling algorithm. Consider already the case where we have p input variables (p could also be just 1).

1. Select randomly $p + 1$ observations. If these observations are in a "general" position (and not in a subspace), they allow for an exact fit of a hyperplane (or line if $p = 1$).
2. Compute the residuals to this hyperplane for all observations, and the value of the LMS criterion.
3. Iterate 1. and 2. "very often". The selected solution is the one with the smallest value of the LMS criterion.

The computation time increases with increasing n, but particularly with increasing p.

4.6 Least Trimmed Squares (LTS) regression

The LTS estimator (Rousseeuw, 1984) minimizes the sum of the smallest squared residuals, i.e. a trimmed sum. Thus, the squared (!) residuals need to be sorted in ascending order,

$r^2_{(1)} \le r^2_{(2)} \le \ldots \le r^2_{(n)}$ and the criterion is

$$(\hat{\alpha}_{LTS}, \hat{\beta}_{LTS}) = \operatorname*{argmin}_{\alpha,\beta} \sum_{i=1}^{h} r^2_{(i)}$$

with $\frac{n}{2} < h < n$. The parameter h also determines the breakdown point of the LTS estimator, which is in between 0% and 50%. Thus, smaller h leads to a higher breakdown point, but also to a lower statistical efficiency of the estimator.

Fast-LTS algorithm: The subsampling algorithm could also be applied for LTS, but here a much faster algorithm has been developed:

1. Select randomly h observations.
2. Use LS regression with these h observations to estimate the parameters. The resulting fit can of course be affected by outliers.
3. Compute the residuals from *all* observations, and sort their squared values.
4. Take those h observations with the same index as the smalles squared residuals from Step 3.
5. Iterate Steps 2.-4. until convergence. This is guaranteed because the sum of the selected h squared residuals never gets bigger.
6. Carry out Steps 1.-5. several times. The approximate LTS solution is the one with the smallest value of the objective function.

R code: The package `robustbase` offers many robust procedures, not only for regression. LTS regression is implemented in the function `ltsReg()`. This also works in case of more than one input variable, see next section.
Another function, `lmrob()` for MM regression, might even be preferable. The main idea is to minimize the sum of $\rho(r_i/\hat{\sigma})$, thus a function ρ applied to the scaled residuals, where the choice of ρ defines the robustness properties. Also the residual scale needs to be estimated robustly. MM regression can be tuned to achieve breakdown point 50% and at the same time high efficiency. Also, the implemented algorithm also works if there are binary or categorical input variables.

4.7 Regression with several input variables

In the more general case we want to predict an output variable y by using several inputs $x_1, x_2, \ldots, x_p$. For example, if the response y refers to something like product quality (in a production process), then there could be several input variables which could be influential for good or bad quality.

We assume again a linear relationship of the inputs with the response, and thus a model of the form:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

The parameter α is again the intercept, the βs are the slope parameters to the inputs, and ε is the (random) error term.

Then we need to collect data, thus observations measured simultaneously for the response and for all inputs, $(y_i, x_{i1}, x_{i2}, \ldots, x_{ip})$ for $i = 1, \ldots, n$. These are used for parameter estimation, and the estimated parameters $\hat{\alpha}$, $\hat{\beta}_1, \ldots, \hat{\beta}_p$ lead to the fitted values

$$\hat{y}_i = \hat{\alpha} + \hat{\beta}_1 x_{i1} + \ldots + \hat{\beta}_p x_{ip}$$

which are on a hyperplane, and to residuals

$$r_i = y_i - \hat{y}_i \quad \text{for} \quad i = 1, \ldots, n.$$

Similar as in the case of a single input variable we would like to keep the residuals "small", and thus we need to use an appropriate criterion to obtain useful parameter estimates. The LS criterion would be natural, with the only drawback of the sensitivity to outliers. Note that we again can have vertical outliers and (bad) leverage points, but this time the leverage points would be outliers in the p-dimensional space of the input variables. It is not at all trivial to identify outliers in higher dimension, since those do not necessarily have to be extreme in one variable.

This is the reason why we have to rely on an appropriate method. Several (but not all) of the previously mentioned robust regression methods can be extended to the case of multiple input variables. The criteria look similar or are even unchanged, but the algorithms gets more complex.

Example: Consider the data set *rice* from the package *rrcov* with subjective evaluations of 105 different kinds of rice. The response y is the overall evaluation, and the input variables are flavor, appearance, taste, stickiness, and toughness. Our linear model should not only be useful to predict the outcome y, but we would also like to know which of the input variables contributes to the outcome, and how.

Let us start with simple linear regression by considering only the x variable *Favor* as a predictor. We compare the results of LS regression with those of MM regression. The corresponding regression lines are shown in Figure 4.3 (left). The estimated regression parameters from both methods are very similar to each other. This means that LS regression works (because there are no strong outliers), and MM regression also works (even though we have "clean" data!). The *R* code is:

```
library(robustbase)
library(rrcov)
data(rice)
attach(rice)

plot(Favor,Overall_evaluation)
r1 <- lm(Overall_evaluation~Favor,data=rice)
abline(r1,col=6)
r2 <- lmrob(Overall_evaluation~Favor,data=rice)
abline(r2,col=3)
legend("topleft",legend=c("LS regression","MM regression"),
       lty=c(1,1),col=c(6,3))
```

Figure 4.3 (right) presents the resulting fits from classical and robust regression by using the two input variables *Favor* and *Appearance*. Again, there is only a little difference in the results of the two methods.

```
mod <- lm(Overall_evaluation~Favor+Appearance, data=rice)
coef(mod)
```

```
(Intercept)       Favor  Appearance
 -0.1385352   0.5041728   0.7237637
mod2 <- lmrob(Overall_evaluation~Favor+Appearance, data=rice)
coef(mod2)
(Intercept)       Favor  Appearance
 -0.1582099   0.4895092   0.7668813
```

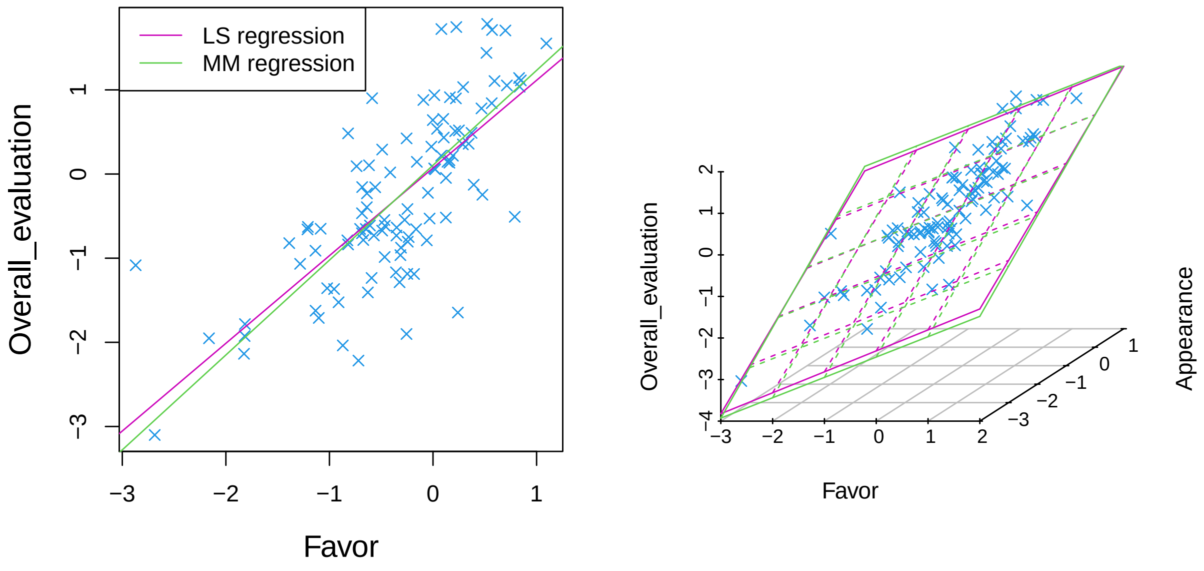


Figure 4.3: LS and MM regression for the rice data set, based on a single explanatory variable (left), and using two explanatory variables (right).

Finally we want to use all 5 input variables in regression. The problem can no longer be visualized. However, we can look at the `summary()` output, here for LS regression:

```
re1 <- lm(Overall_evaluation~.,data=rice)
summary(re1)
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.13026    0.03507  -3.715 0.000337 ***
Flavor       0.19359    0.05398   3.586 0.000523 ***
Appearance   0.10829    0.05993   1.807 0.073805 .
Taste        0.53905    0.08163   6.604 2.02e-09 ***
Stickiness   0.40599    0.07146   5.682 1.34e-07 ***
Toughness    0.03513    0.05733   0.613 0.541460
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2745 on 99 degrees of freedom
Multiple R-squared:  0.9306,Adjusted R-squared:  0.927
F-statistic: 265.3 on 5 and 99 DF,  p-value: < 2.2e-16
```

Under `Estimate` we can find all estimated regression parameters. The remaining columns are used for hypothesis tests: we test if the regression parameter is zero. The last column is the p-value, and if $p < 0.05$ we can reject the null hypothesis of a zero coefficient, implying that the variable contributes to explaining the response. So, now we know that *Flavor*, *Taste* and *Stickiness* positively contribute to the *Overall_evaluation*.

The multiple R-squared is a measure of fit, it is the squared correlation of the response with the fitted values. Thus the model seems to fit quite well, which hopefully means that it also predicts well.

For MM regression we obtain the following results:

```
re2 <- lmrob(Overall_evaluation~.,data=rice)
summary(re2)
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.09841    0.03729  -2.639  0.00966 **
Flavor       0.21569    0.07926   2.721  0.00769 **
Appearance   0.02917    0.07986   0.365  0.71572
Taste        0.60889    0.09639   6.317 7.64e-09 ***
Stickiness   0.36465    0.07954   4.584 1.33e-05 ***
Toughness    0.01428    0.04812   0.297  0.76726
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Robust residual standard error: 0.2131
Multiple R-squared:  0.954,Adjusted R-squared:  0.9517
Convergence in 18 IRWLS iterations

Robustness weights:
 observation 75 is an outlier with |weight| = 0 ( < 0.00095);
 6 weights are ~= 1. The remaining 98 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0154  0.8960  0.9506  0.8880  0.9795  0.9989
```

The outcome seems to be quite similar to LS, with the exception of the coefficient for *Appearance*; also the p-value is much bigger.

Here we also get information about the outlyingness of the observations in terms of *Robustness weights*, which are small in case of outliers. Accordingly, observation 75 is an outlier.

Figure 4.4 shows diagnostics plots for MM regression which are obtained by `plot(re2)`. The left plot shows $\hat{y}_i$ against y_i. The values are very similar to each other, which was already suggested by the high value of the multiple R-squared. The outlier 75 deviates slightly. The right plot presents a robust distance measure (Mahalanobis distance) in the space of the explanatory variables on the horizontal axis. Distances larger than the indicated cutoff value would refer to leverage points. The vertical axis is for the standardized residuals, again with cutoff values that help to separate regular residuals from vertical outliers. Observation 75, for example, is a vertical outlier. There are a few leverage points, but they are all good leverage points.

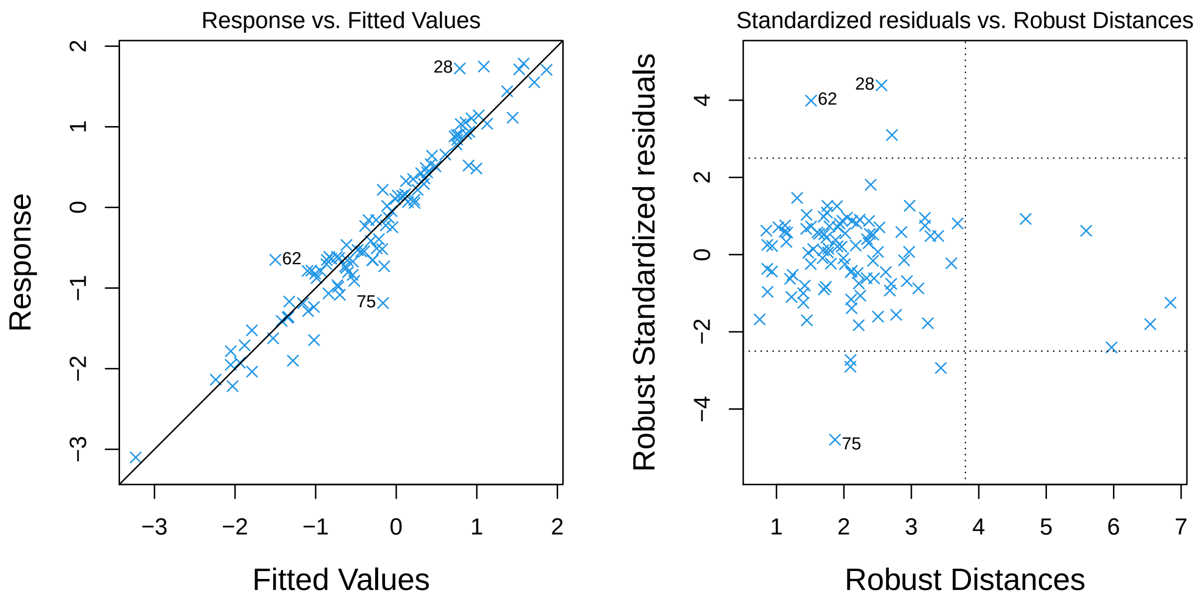


Figure 4.4: Diagnostics plots for robust MM regression based on the rice data set.

Chapter 5

Estimation of non-linear trends

We assume pairs of observations $(x_1, y_1), \ldots, (x_n, y_n)$. In this setting, the x-values could also refer to time points, and the y-values to the observed information over time.

The following sections treat the problem of smoothing a signal, and thus the problem of estimating a non-linear trend. Also trend estimation can be influenced by outliers, and thus a focus is on robust non-linear trend estimation. Some methods also require equidistant measurements, typically along the time axis, while for other methods this is not important.

5.1 Non-linear smoothing for equidistant information

Here we assume to have time series values x_t, measured at equidistant time points $t = \ldots$, -2, -1, 0, 1, 2, …, i.e. we have equal differences between the time points. Our goal is to smooth the time series by making use of simple exploratory techniques which are to a certain extent also robust against artifacts in the signal (peaks, jumps). The smoothed signal should make patterns in the time series visible.

The main idea is to use an "algorithm" S for smoothing the time series values x_t. When applying S on x_t we obtain the smoothed values z_t. Thus:

time series	=	smoothed signal	+	residuals
x_t	$=$	$S(x_t)$	$+$	r_t
x_t	$=$	z_t	$+$	r_t

$$r_t = x_t - S(x_t)$$

Linear filter: This is a very simple idea: for smoothing at time point t we take a weighted sum of the time series values in a local neighborhood around t, where the weights α_i have to be defined appropriately:

$$z_t = \sum_{i=-l_1}^{l_2} \alpha_i x_{t+i}$$

with $0 \leq l_1$, $0 \leq l_2$, $\alpha_{-l_1} \neq 0$, $\alpha_{+l_2} \neq 0$.

An example for such an algorithm is: $z_t = \frac{1}{4}x_{t-1} + \frac{1}{2}x_t + \frac{1}{4}x_{t+1}$

A linear filter can be very sensitive to outliers. An idea towards more robustness is to simply use medians as alternative to (weighted) means. In either case, there are some issues that need consideration:

- In many algorithms, the window of time points used for smoothing is centered at the time point where smoothing is done. The **span** s is defined as the number of time points to the farthest end of this window. The **window width** w is the number of time points covered by this window.
- Smoothing by the median with an odd number of $w = 2s + 1$:

  $$z_t = S(x_t) = \text{median}(x_{t-s}, x_{t-s+1}, \ldots, x_t, x_{t+1}, \ldots, x_{t+s})$$

  This results in the smoothed time series $\ldots, z_{t-1}, z_t, z_{t+1}, \ldots$.

  An example is: $s = 1$: $z_t = \text{median}(x_{t-1}, x_t, x_{t+1})$

- Smoothing by the median with an even number of $w = 2s$:

  $$z_{t+\frac{1}{2}} = S(x_t) = \text{median}(x_{t-s+1}, \ldots, x_t, x_{t+1}, \ldots, x_{t+s})$$

  This results in the smoothed time series $\ldots, z_{t-1-\frac{1}{2}}, z_{t-\frac{1}{2}}, z_{t+\frac{1}{2}}, z_{t+1+\frac{1}{2}}, \ldots$

  An example is: $s = 1$: $z_{t+\frac{1}{2}} = \text{median}(x_t, x_{t+1})$

  If we want to have the smoothed values at the original time points, we can apply again median smoothing with an even number w, not necessarily the same as before: $z_t = S_2(S_1(x_t))$

- **Extrapolation:** It is usually desirable that the smoothed signal has the same number of time points as the original signal, but median smoothing would lose time points on both ends.

  Assume a time series $x_1, x_2, \ldots, x_{T-1}, x_T$. The best what we can achieve with median smoothing is a smoothed signal $z_2, z_3, \ldots, z_{T-2}, z_{T-1}$, possibly by applying median smoothing on both ends with small w.

  Then the idea is to use linear extrapolation of z_2 and z_3 to obtain a value z_0. The result is $z_0 = 3z_2 - 2z_3$. Similarly, we obtain $z_{T+1} = 3z_{T-1} - 2z_{T-2}$. Then we use median filtering as follows:

  $$z_1 = \text{median}(3z_2 - 2z_3, x_1, z_2)$$
  $$z_T = \text{median}(3z_{T-1} - 2z_{T-2}, x_T, z_{T-1})$$

Figure 5.1 (left) shows a time series with body temperatures of a cow. The temperature has been measured at 6:30am on 75 consecutive days. The right plot shows the result of different smoothing algorithms. Median smoothing with the *R* function `smooth()`, here with $w = 3$, median smoothing with `med.filter()` from the package `robfilter`, with $w = 5$, and mean smoothing with `runmean()` from package `caTools` with $w = 3$. For the latter we see some sensitivity to the outlier at the beginning of the series.

5.2 Robust smoothing and outlier filtering ("robfilter")

Non-linear robust smoothing will produce a smoothed signal which is hopefully also not much affected by outliers. Thus, in a second step one can inspect the residuals in order to identify potential outliers. This second step is called *filtering*.

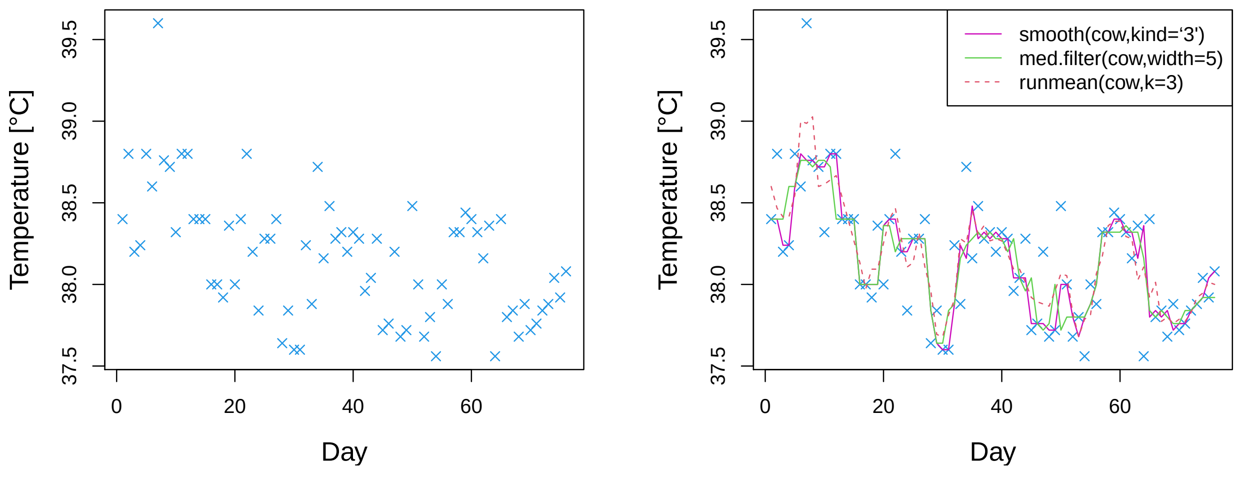


Figure 5.1: Time series with body measurements of a cow, and application of different smoothing algorithms.

Even after simple median based smoothing it would be possible to do outlier filtering. However, there is one difficulty: In the linear (regression) case we assumed that the residual variance is constant. Thus, the residual variance can be estimated from all (non-outlying) residuals, we can scale the residuals, and look if the scaled residuals exceed the interval ± 2 or ± 2.5, see Figure 4.4 (right). With non-linearity we can no longer assume constant residual variance! Rather, the residual variance needs to be estimated locally (and robustly). It is unclear how to do this with simple median smoothing.

Nice solutions to this problem are implemented in the package `robfilter`. The package contains several algorithms for robust smoothing and filtering, also in online settings. Especially with larger amounts of data (e.g. signals in an intensive care unit), online versions need to be quick to compute and reliable for outlier diagnostics.

In the following we will discuss such an algorithm. It makes use of our good friend, the repeated median from Section 4.4. We assume a time series with values x_t, and equidistant time points $t = 1, \dots, T$. We also assume a window width $w = 2s + 1$, with $s > 0$. Thus, for smooting a value x_t (filtering at time point t) we will make use of the values $\{x_{t-s}, x_{t-s+1}, \dots, x_t, x_{t+1}, \dots, x_{t+s}\}$.

Let us assume an underlying but non-observable signal μ_t, for $t = 1, \dots, T$, which we want to estimate based on the given data. In fact, the observed data are generated according to the model $x_t = \mu_t + \epsilon_t$. Here, ϵ_t is an error, which is considered as a mixture of a normal distribution and a "heavy-tailed" (outlier generating) distribution.

A crucial assumption is (approximative) **local linearity** of μ_t within the time window $2s+1$:

$$\mu_{t+i} \approx \mu_t + \beta_t \cdot i \qquad \text{for } i = -s, -s+1, \dots, s.$$

The parameters of this linear function are the "level" μ_t and the slope β_t, which neeed to be estimated. One could use any robust regression method for this purpose. The authors proposed to use the Repeated Median estimator:

$$\begin{aligned}
\hat{\beta}_t &= \underset{-s \le i \le s}{\mathit{median}} \left(\underset{\substack{-s \le j \le s \\ j \ne i}}{\mathit{median}} \left(\tfrac{x_{t+i} - x_{t+j}}{i-j} \right) \right) \\
\hat{\mu}_t &= \underset{-s \le i \le s}{\mathit{median}} \left(x_{t+i} - \hat{\beta}_t \cdot i \right)
\end{aligned}$$

For the outlier diagnostics we need an estimation of the residual scale σ_t at *every* time point

t. This can be obtained with the above principle of local linearity. In the time window around t we obtain the residuals

$$r_t(t+i) = x_{t+i} - \hat{\mu}_t - \hat{\beta}_t \cdot i \quad \text{for } i = -s, -s+1, \ldots, s.$$

We can use any robust scale estimator to estimate the residual scale σ_t at time point t, such as the Q_n estimator, see Section 2.2:

$$\hat{\sigma}_t = 2.219 \cdot \{|r_t(t+i) - r_t(t+j)|; i < j\}_{(k)} \text{ with } k = \binom{h}{2} \text{ and } h = \lfloor (2s+1)/2 \rfloor + 1.$$

Based on this scale estimation we can use the standard procedure for outlier diagnostics:

An outlier is identified if $|\hat{r}_t| > 2 \cdot \hat{\sigma}_t$, where $\hat{r}_t = x_t - \hat{\mu}_t$ is the estimated residual value at time point t.

Example: Consider again the cow data set from the last section. The *R* code below shows how to apply the described algorithm. Here the window width was set to $2s+1 = 7$ (it needs to be an odd number). The result is shown in Figure 5.2 left. The right plot shows the result for window width $2s+1 = 19$. The result heavily depends on this parameter. The package `robfilter` also contains an algorithm (`scarm.filter()`) which automatically adjusts the window width to the given data, based on statistical tests. However, this algorithm needs a lot more data.

```
library(robfilter)
res <- robust.filter(cow,width=7) # trend by RM, scale by Qn
plot(res)
res
 # $level
 # [1] 38.4 38.48667 38.57333 38.66 38.47733     ...
 # $slope
 # [1] 0.086667 0.086667 0.086667 0.086667 0.122667     ...
 # $sigma
 # [1] 0.830133 0.830133 0.830133 0.830133 0.833152     ...

# indicate outliers:
ind <- which(res$ol!=0)
points(ind,cow[ind])
```

5.3 Local polynomial regression fitting ("loess")

This procedure assumes data pairs (x_i, y_i), for $i = 1, \ldots, n$, and it is not typically applied in a time series context (but it could). This also means that we do not assume equidistant points along x, and we would like to smooth along y to obtain smoothed $\hat{y}$ values.

The key idea is to use locally weighted regression, where the regression model is either

linear: $y = \alpha + \beta x + \varepsilon$

or

quadratic: $y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$.

In either case, the regression coefficients are estimated by a weighted residual sum of squares, where the weights consider the local aspect (in x direction) and robustness aspects (in y direction.

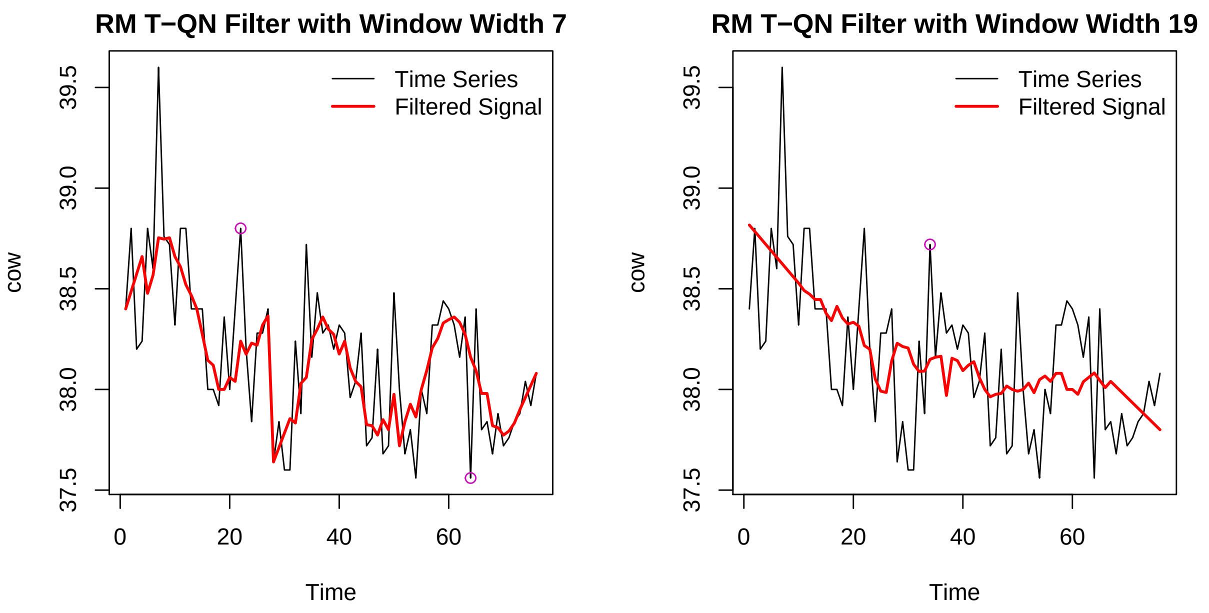


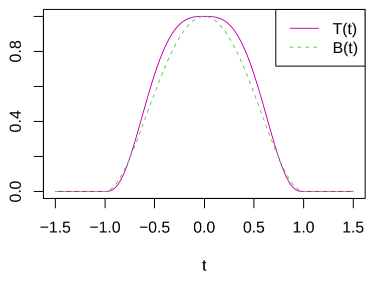
Figure 5.2: Application of robust filtering to the cow data, with window width 7 (left) and 19 (right). Identified outliers are indicated.

The algorithm is described below, here for the linear model. We fix a parameter s, called *span*, defining the proportion of data points to be used for local smoothing. A default parameter is $s = 0.75$. Thus, $q := \lfloor ns + 0.5 \rfloor$ is the resulting number of data points. Further, define the weight functions

$$\text{tri-cube function } T(t) := \begin{cases} (1- \mid t \mid^3)^3 & \mid t \mid < 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\text{biweight function } B(t) := \begin{cases} (1-t^2)^2 & \mid t \mid < 1 \\ 0 & \text{otherwise.} \end{cases}$$



We want to smooth at index i, thus in a local neighborhood of (x_i, y_i), to obtained the smoothed value $\hat{y}_i$.

1. Compute all distances to x_i along the x-coordinate:

   $$d_{ik} := \mid x_i - x_k \mid \text{ for } k = 1, \ldots, n.$$

   Denote d_{i_q} as the q-th biggest distance.

2. Compute weights according to distance in x-coordinate:

   $$t_i(x_k) := \begin{cases} T\left(\frac{d_{ik}}{d_{i_q}}\right) & d_{i_q} \neq 0 \\ 1 & d_{i_q} = 0 \end{cases} \qquad \text{for } k = 1, \ldots, n \ .$$

3. Weighted LS regression:

   $$(\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}) = \underset{\alpha,\beta}{\operatorname{argmin}} \sum_{k=1}^{n} t_i(x_k)(y_k - \alpha - \beta x_k)^2$$

   This results in the smoothed value at index i: $\hat{y}_i = \hat{\alpha}^{(i)} + \hat{\beta}^{(i)} x_i$

4. Repeat Steps 1.-3. for all $i = 1, \ldots, n$.
5. Compute residuals $r_i = y_i - \hat{y}_i$, for $i = 1, \ldots, n$.
   Estimate robustly the residual scale $\hat{\sigma}$.
6. Compute weights for the size of the scaled residuals:

   $$w(x_k) := B(\tfrac{r_k}{2\hat{\sigma}}) \qquad \text{for } k = 1, \ldots, n,$$

   and use them in Step 3. for weighted LS regression, thus the weights are $w(x_k)t_i(x_k)$.
   Repeat all these steps several times.

Example: This algorithm is implemented (with some more functionality) as function `loess()`. Here is an application to the cow temperature data set:

```
df <- data.frame(time=seq(from=1,to=length(cow)),Temperature=cow)
plot(df)
res1 <- loess(Temperature~time, data=df)
pred1 <- predict(res1)
lines(df$time,pred1)
```

We have used the default settings of the function, in particular the default of *span s*, which is 0.75. This produces the smoothed pink line in Figure 5.3 (left). When using a smaller *span* of 0.3, we obtain the green smoothed curve, which focuses more on the local behavior of the data. Both curves seem to be informative.

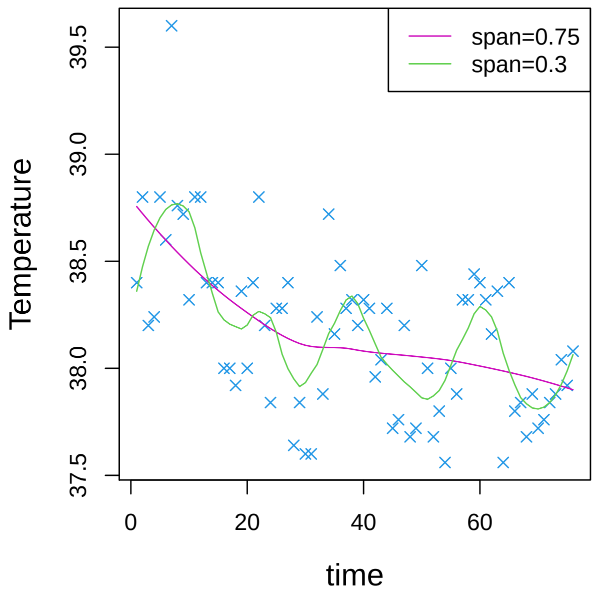
The `loess()` function can also be used for extrapolation. In that case, the function needs to be applied as follows:

```
res2 <- loess(Temperature~time, data=df,
              control = loess.control(surface = "direct"))
timeext <- seq(from=1,to=100,by=1)
pred2 <- predict(res2,data.frame(time=timeext),se=TRUE)
plot(df,xlim=c(0,100),ylim=c(36.5,40))
lines(timeext,pred2$fit,col=3)
lines(timeext,pred2$fit+2*pred2$se.fit,lty=2,col=3)
lines(timeext,pred2$fit-2*pred2$se.fit,lty=2,col=3)
```

Figure 5.3 presents the outcome, again for both values of *span*. The time has been extended up to 100 days, and the extrapolation gets very sensitive if *span* gets smaller. The estimated standard errors are shown in addition as bands around the smoothed lines, but they seem not quite useful for the extrapolation for the *span* 0.3.

The `loess()` smoother is also implemented in a version of the scatterplot matrix, and it can be useful to see the trend in each of the scatterplots. Most importantly, the scatterplot matrix shows for all pairs of variables both versions (i.e. x against y, and y against x). When exchanging the axes, the outcome of the smoothed values are in general different.

Example: The data set `environmental` of the package `lattice` contains daily measurements of ozone concentration, wind speed, temperature and solar radiation in New York City, from May to September of 1973. Figure 5.4 shows the scatterplot matrix of the data, together with the smoothed lines from `loess()`. This plot can be produced with the function `scatterplotMatrix()` from the package `car`.



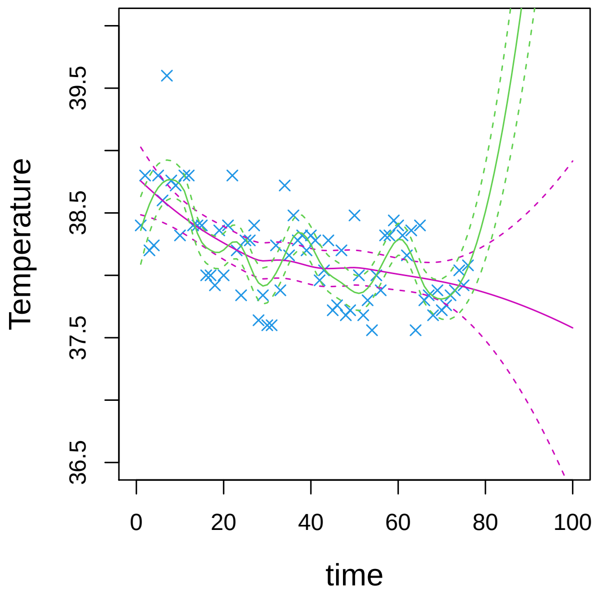


Figure 5.3: Smoothing with `loess()`: The left plot shows the smoothed curves by using two different values for *span*, and the right plot shows extrapolations together with standard error bands.

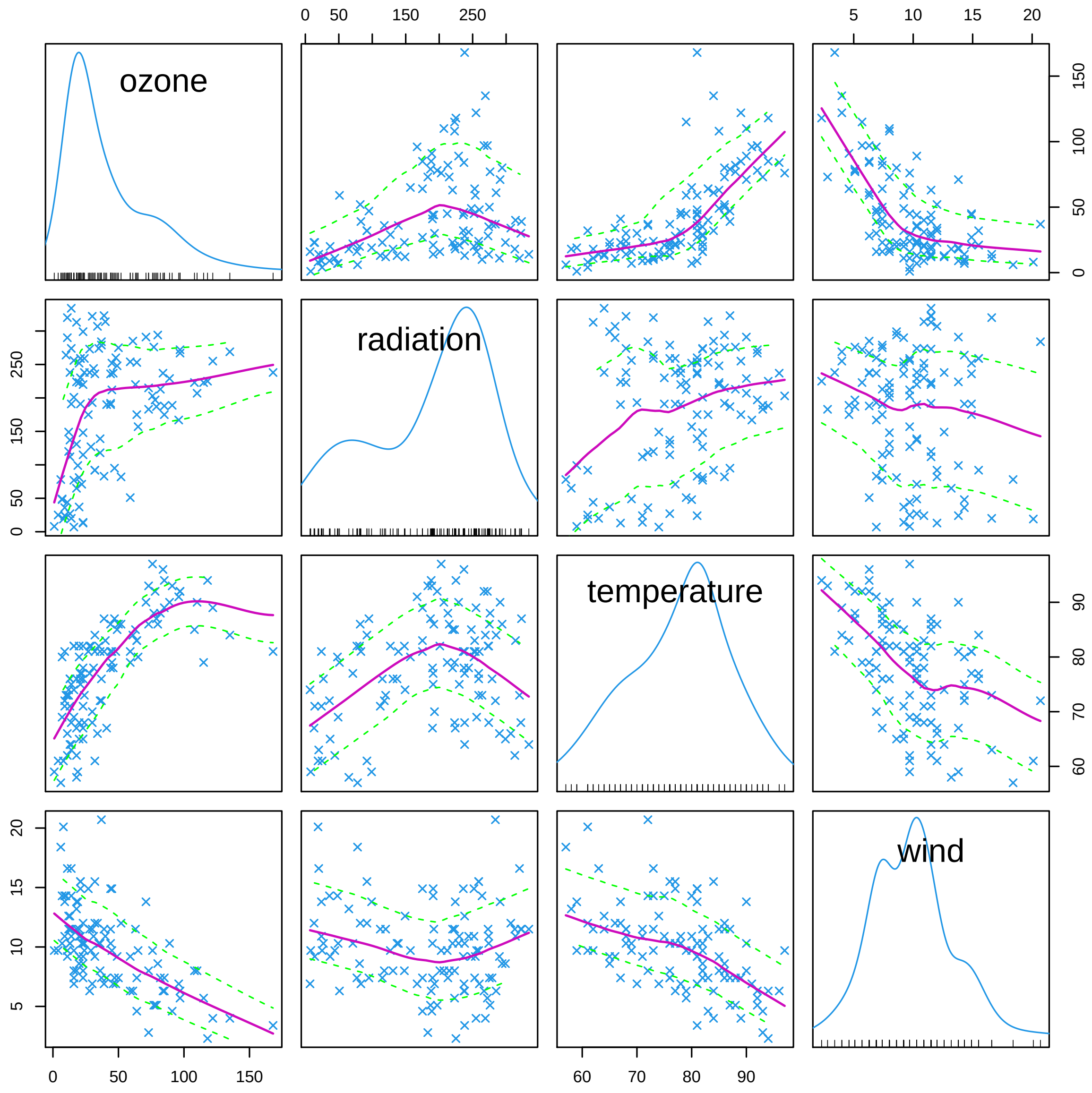


Figure 5.4: Scatterplot matrix of environmental data, with `loess()` smoothed lines.

Chapter 6

Time series analysis – an introduction

The goal of this chapter is to provide an overview of different approaches to analyzing data that depend on time. In previous chapters we commonly assumed that the observations are independent from each other – this is different here, and the form of the dependency is also crucial for the way how we analyze the data. Some important questions in this context are:

- Are there trends or seasonal changes? If so, how can they be quantified?
- Is there still remaining structure in the residuals, which are defined as signal minus trend and seasonality?
- Is the time series following a certain pattern over time which we could try to model?
- Is it possible to do forecasting (predict into the future)?
- Are there structural breaks in the time series, or outliers?

We will not cover all these topics, but provide insight into some of them. Here we will only treat *univariate time series*, thus values x_t measured for one variable over time points $t = 1, \ldots, T$, which we assume to be equidistant.

As an example, consider a time series of the monthly produced amount of beer in Australia, in the time period January 1956 until August 1995. The data are available in the package `tsapp` as `data(BEER)`. This is already a time series object. If this would not be the case, one could create such an object with `ts(mydata,start=1956,freq=12)`. Time series objects can be visualized easily with:

```
plot(BEER)
```

Figure 6.1 (top) shows the time series graphically. We can see a clear trend, but also clear seasonal patterns. A more detailed look at the data shows that the deviations towards higher values are somewhat bigger than those towards lower values. Thus, the deviations seem not to be symmetric around the mean, but rather skewed. This could be a disadvantage later on once we want to do estimations. A simple way out is to take the **logarithm** of the data (log-transformation). The result is shown in the bottom plot of Figure 6.1. Now the local deviations seem to be more symmetric. In fact, the differences of neighboring values in the log-transformed time series are:

$$\ln(x_t) - \ln(x_{t-1}) = \ln\left(\frac{x_t}{x_{t-1}}\right) \approx \frac{x_t - x_{t-1}}{x_{t-1}}$$

The last relationship holds if the neighboring values are (very) similar, which should be true in the time series context. Thus, with log-transformed values we are interested in *relative* *differences*, and not in *absolute* ones. Also in other applications, such as for financial time series, a log-transformation is very common; the terminology they are using is *log-returns*. In the following we will work with these log-transformed values.

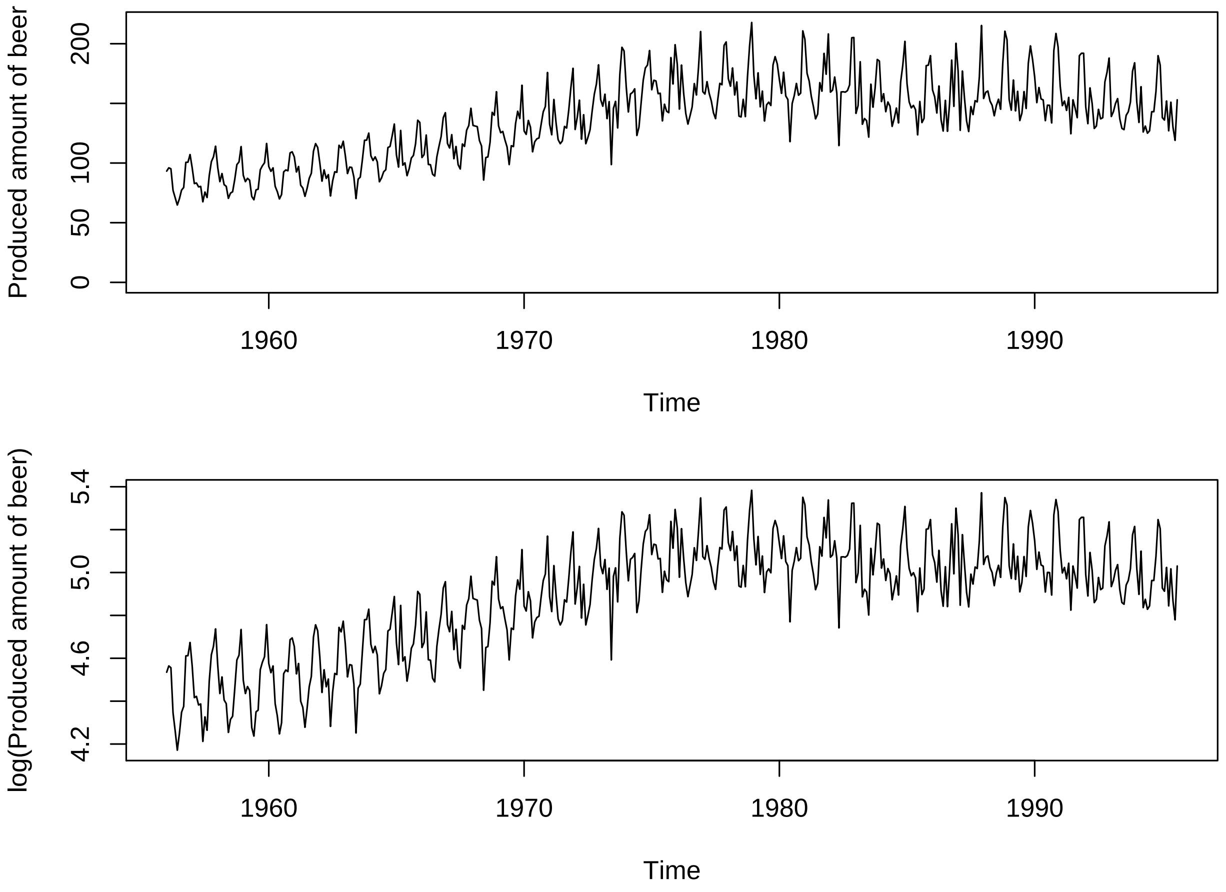


Figure 6.1: Original (top) and log-transformed (bottom) time series of the beer data.

6.1 Decomposition of time series into components

The previous example has shown trends and seasonality. A **trend** means that the time series increases or decreases for a longer period of time. **Seasonality** refers to certain regular variations, such as regular changes for the same months over the years. Often we observe both trend and seasonality at the same time, but we are still interested in estimating these characteristics separately. Thus, we want to decompose the time series into the following components:

$$x_t = \tau_t + \delta_t + e_t \quad \text{for } t = 1, \ldots, T,$$

with the *trend component* τ_t, the *seasonal component* δ_t, and the *remainder or error component* e_t. Assume that we also know the periodicity P of the time series. For the beer time series we have $P = 12$ because we have monthly data. Since the length of the time series is T, we have $C = \lfloor T/P \rfloor$ cycles.

In Chapter 5 we already had mentioned methods for smoothing time series signals, and for estimating non-linear trends. The `R` function `stl` makes use of such methods, in particular of the function `loess`, see Section 5.3. The method `stl` works according to the following

scheme:
Assume that we are in iteration k, for $k = 1, 2, \ldots$, and we already have estimates of $\tau_t^{(k)}$ and $\delta_t^{(k)}$. Then iteration $(k+1)$ is as follows:

(1) Eliminating the trend (*detrending*): $x_t - \tau_t^{(k)}$

(2) `loess` applied to (1) for the time points $t_i = t + i \cdot P$, with $i = 0, \ldots, C-1$, successively for all $t \in \{1, \ldots, P\}$. For example, we smooth all values for January, then all values for February, etc.

(3) Application of a (linear) filter on the values from (2), see Chapter 5, yields $\delta_t^{(k+1)}$.

(4) Eliminating seasonality: $x_t - \delta_t^{(k+1)}$

(5) `loess` applied to (4) yields $\tau_t^{(k+1)}$

Similar to the *loess* algorithm, see Section 5.3, we compute weights for the remainder $e_t^{(k+1)} = x_t - \tau_t^{(k+1)} - \delta_t^{(k+1)}$ (resuduals), with the goal to downweight outliers within the iterations of the above procedure.

Figure 6.2 tries to provide more insights into this procedure. The plot on top shows the detrended time series, thus the result after Step (1). The bottom plot collects all monthly time points in order to apply Step (2). This plot is also called *cycle plot*, available in `R` as `monthplot()`.

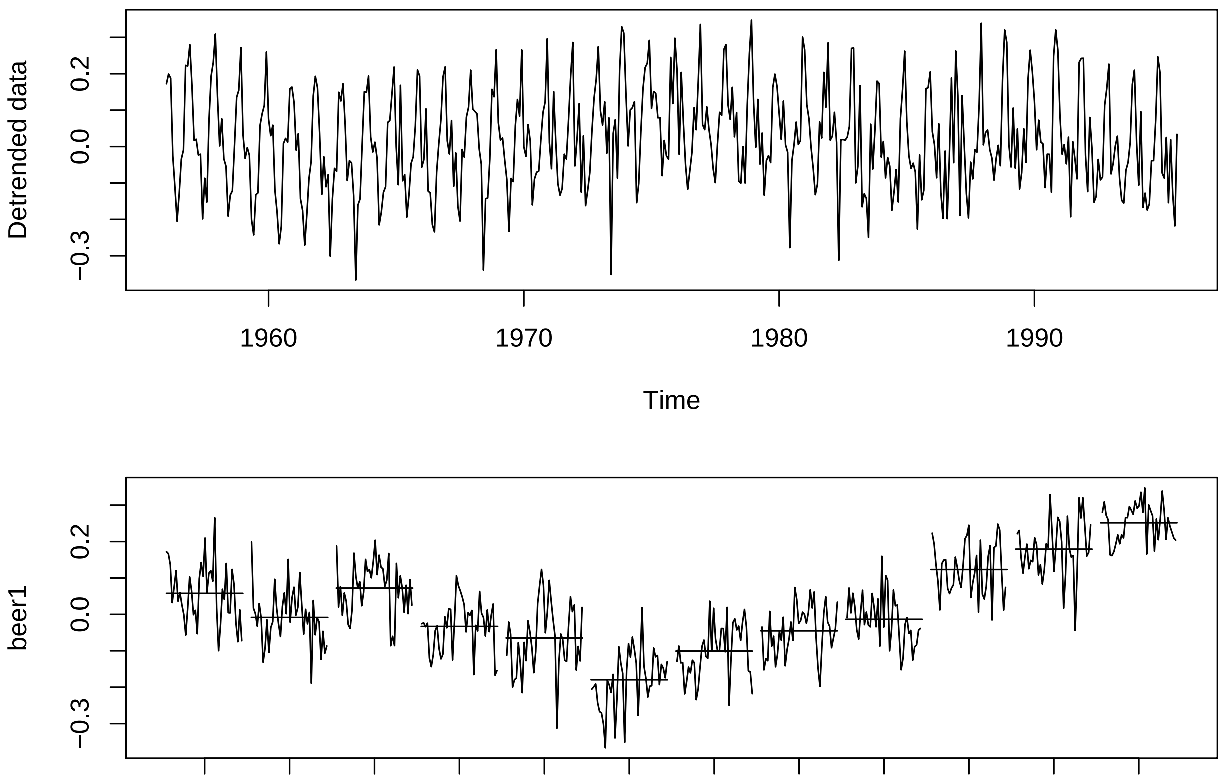


Figure 6.2: Detrended time series (top) and cycle plot (bottom), collecting the single months of the time series of the beer data.

Figure 6.3 shows the result of `stl` applied to the log-transformed beer time series. The R code is:

```
plot(stl(log(BEER),s.window="periodic"))
```

If the remainder would not contain any further structure (correlation structure, temporal patterns, etc.), then we would talk about “white noise”. If this is not the case, the remainder would still contain valuable information which should be modeled appropriately. The results in Figure 6.3 do not reveal clear patterns for the remainder, but still some “peaks” of varying magnitude which could be of interest to the modeler.

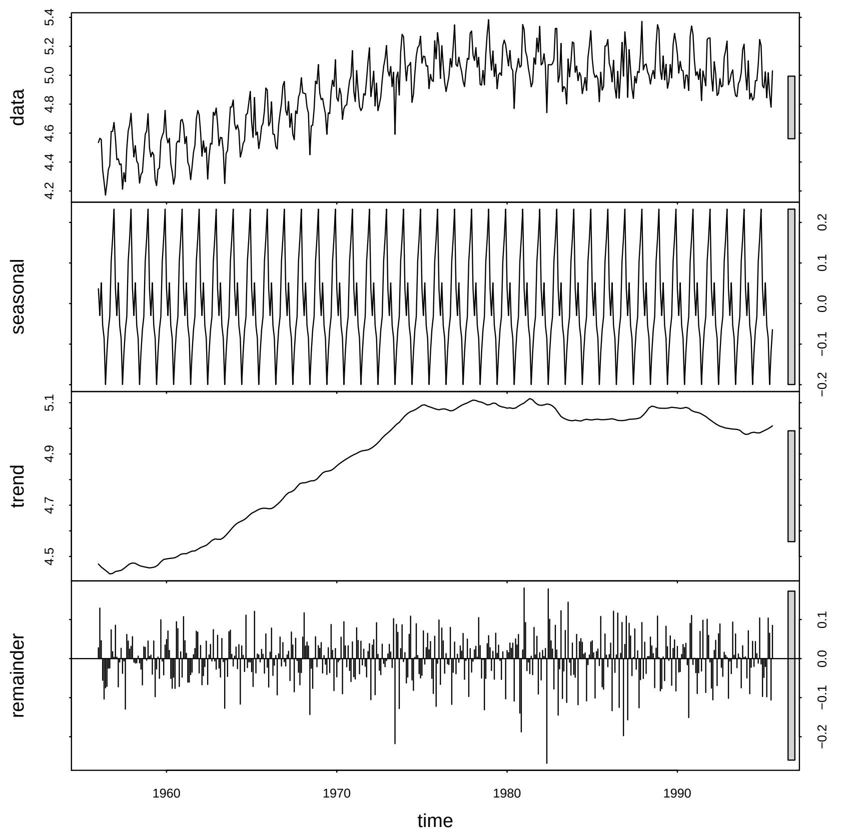


Figure 6.3: Decomposition of the time series into its single components. (`stl`)

The next section shows some attempts to modeling time series which are **not recommended**.

6.2 Regression model for time series

One important goal when modeling time series is forecasting future values. We could now try to use standard regression models for this purpose. Some simple models will be applied below; however, they do not appropriately make use of the time dependency of the data.

6.2.1 Linear model

In Chapter 4 we discussed methods for (robustly) estimating linear trends. A linear model for log-transformed time series is of the form

$$\ln(x_t) = \beta_0 + \beta_1 t + e_t,$$

with the coefficients β_0 and β_1, and the error term e_t. The coefficients can be estimated with the methods from Chapter 4. In the following we will use LTS regression.

For our beer data time series we would proceed as follows:

```
t <- (1:length(BEER))/12+1956    # construct time axis
logBEER <- log(BEER)             # model log-transformed values
library(rrcov)                   # for LTS regression
plot(logBEER)
res <- ltsReg(logBEER~t)         # LTS regression with linear model
lines(t,res$fit)
```

Figure 6.4 (left) shows the result. The linear model is of course too simple, and thus we should proceed with a more complex model.

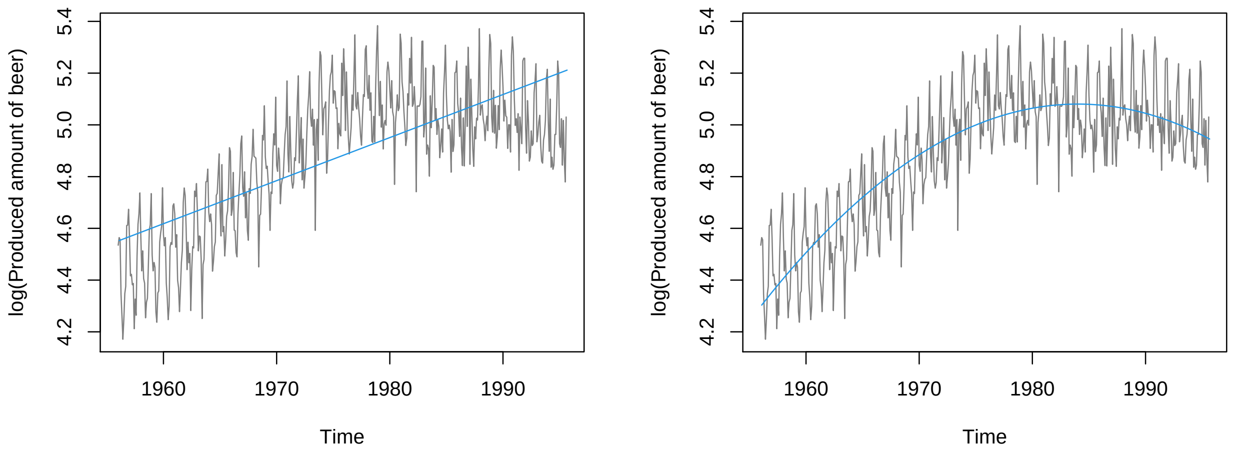


Figure 6.4: Linear model (left) and model with a squared term (right) for the beer time series.

6.2.2 Regression with a quadratic term

The linear model from above can be easily extended with a quadratic term:

$$\ln(x_t) = \beta_0 + \beta_1 t + \beta_2 t^2 + e_t$$

For our exaple this can be realized as follows:

```
t2 <- t^2                        # quadratic term
plot(logBEER)
res <- ltsReg(logBEER~t+t2)      # LTS regression for new model
lines(t,res$fit)
```

The result is presented in Figure 6.4 (right). We obtain a smooth estimation of the trend, and this could already be useful for a simple forecast of future values. Denote $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ as the estimated regression parameters, then

$$\hat{x}_t = \exp\left(\hat{\beta}_0 + \hat{\beta}_1 t + \hat{\beta}_2 t^2\right)$$

is the forecast for future time points $t = T+1, \ldots, N$. However, since the model is still relatively simple, the forecast will only be of limited use, and in the best case just applicable to value in the near future.

6.2.3 Regression with Fourier coefficients

We could represent a time series signal $f(t)$ with periodicity P also with a (finite) Fourier series:

$$f(t) = a_0 + \sum_{j=1}^{J} \Big(a_j \cos(\omega_j t) + b_j \sin(\omega_j t) \Big), \qquad \text{with } \omega_j = \frac{2\pi j}{P}.$$

The terms in the series correspond to frequencies with higher oscillation, starting with the first harmonic with periodicity P. If we would include in our model also the first term, we would be able to model seasonality. Then we would have:

$$\ln(x_t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 \cos\left(\frac{2\pi}{P} \cdot t\right) + \beta_4 \sin\left(\frac{2\pi}{P} \cdot t\right) + e_t.$$

For our example this works as follows (we already divided `t` by $P = 12$ previously!):

```
cos.t <- cos(2*pi*t)
sin.t <- sin(2*pi*t)
plot(logBEER)
res <- ltsReg(logBEER~t+t2+cos.t+sin.t) # LTS regression for new model
lines(t,res$fit)
```

Figure 6.5 shows the result, which is still far from being "perfect" for forecasts. We could still include terms of higher order to improve the fit, which meeans that we have to estimate even more parameters.

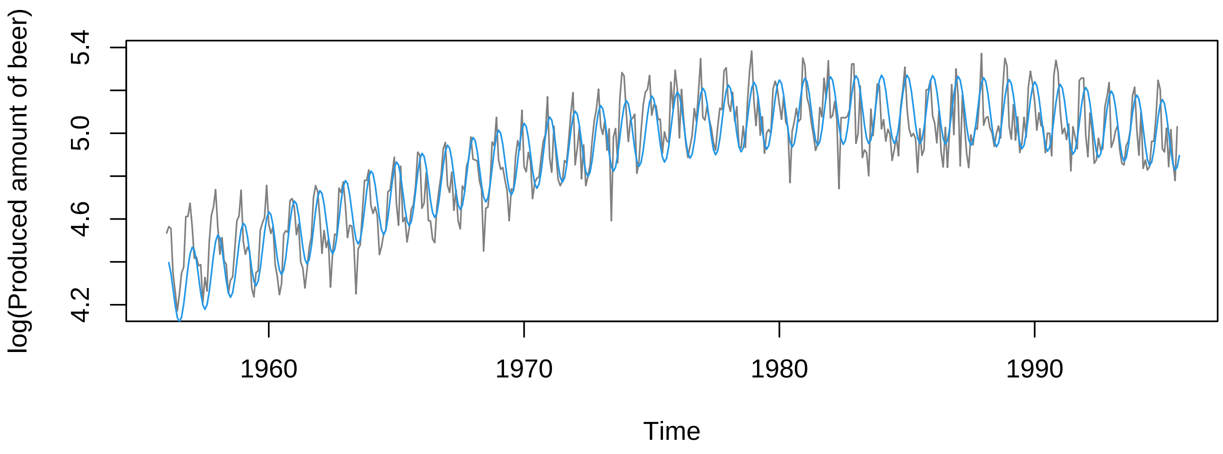


Figure 6.5: Model with Fourier coefficients for the beer time series.

We can obtain information about statistical inference in R with `summary(res)` for our model:

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
Intercept -4.161e+03  1.411e+02 -29.480   <2e-16 ***
t          4.198e+00  1.429e-01  29.387   <2e-16 ***
t2        -1.058e-03  3.615e-05 -29.260   <2e-16 ***
cos.t      1.582e-01  6.014e-03  26.308   <2e-16 ***
sin.t      7.136e-03  5.966e-03   1.196    0.232
```

This shows that all coefficients except β_4 are significantly different from 0. However, since we have a representation with Fourier coefficients, both the sine and the cosine should be used in the model, and they should not be isolated from each other.

6.3 Exponential smoothing

The forecast of future values could be done by weighting the measured time series values according to their history: More recent values could receive a higher weight, while values that are further away in the past receive small weight and thus have only little contribution to the forecast.

Assume observed time series values x_t for $t = 1, \ldots, T$. *Exponential smoothing* provides smoothed values at every time point, but it also allows to obtain forecasts at least into the near future. The smoothed values are denoted with the symbol $\tilde{}$, and in particular the smoothed value $\tilde{x}_t$ at time point t is obtained by

$$\tilde{x}_t = \alpha x_t + (1-\alpha)\tilde{x}_{t-1},$$

with a smoothing parameter $0 < \alpha < 1$. Thus, the smoothed value at t depends on the actual value and on the smoothed value from time point $t-1$, with appropriate weights. The smaller α is chosen, the smaller is the contribution of the most recent values, which implies that the sequence of the values $\tilde{x}_t$ gets smoother. The starting value $\tilde{x}_m$ can be taken as the arithmetic mean of the first m values of x_t.

In order to select the smoothing parameter α, one can take a criterion based on the sum of squared residuals,

$$\sum_{t=m}^{T} (x_t - \tilde{x}_{t-1})^2 \longrightarrow \min$$

but also more robust criteria are possible.

Since the model can be applied also to the previous time point,

$$\tilde{x}_{t-1} = \alpha x_{t-1} + (1-\alpha)\tilde{x}_{t-2},$$

it is immediate to see that this weighting scheme results in a recursive computation:

$$\tilde{x}_t = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + \alpha(1-\alpha)^3 x_{t-3} + \ldots$$

The weights into the past decrease *exponentially*, which justifies the name of the method.

Now we look at the problem of forecasting. Suppose we are at time point t, and we would like to forecast h time points into the future (we talk about a horizon h). This forecast is denoted by $\hat{x}_{t+h|t}$, and since we have not more information, we can only use the actual smoothed value for this purpose, thus

$$\hat{x}_{t+h|t} = \tilde{x}_t.$$

The forecast of all future values thus does not depend on the horizon h, for which we would like to have our prediction. This forecast is useful for *stationary time series*, which neither have trend nor seasonality, and which go back to their average quickly. It is not useful for time series with trend (or seasonality).

The so-called **Holt-Winters** method includes a trend variable b_t:

$$\tilde{x}_t = \alpha x_t + (1-\alpha)(\tilde{x}_{t-1} + b_{t-1})$$
$$b_t = \beta(\tilde{x}_t - \tilde{x}_{t-1}) + (1-\beta)b_{t-1}$$

At every time point t, the parameter b_t estimates the local slope of the time series. The parameters α and β are again in the interval $(0, 1)$, and they regularize the degree of amoothing.

The forecast with Holt-Winters for a horizon h is given by

$$\hat{x}_{t+h|t} = \tilde{x}_t + hb_t.$$

Thus, the forecast is done by a linear equation, with the most recent slope parameter b_t, starting from $\tilde{x}_t$.

Similarly, one could also include a seasonal component.

For the beer time series we can use the method of Holt-Winters in R as follows:

```
plot(BEER)
BEER.hw <- HoltWinters(BEER)        # Holt-Winters
lines(BEER.hw$fitted[,1])           # smoothed line
```

Here we consider both trend and seasonal component. The result of the smoothing method is shown in Figure 6.6 (top). The estimated parameters are:

```
Smoothing parameters:
 alpha:  0.07532444
 beta :  0.07434971                 # parameter for the trend
 gamma:  0.143887                   # parameter for seasonality
```

With this model we can now do forecasting, in our case for the next 48 months:

```
plot(BEER,xlim=c(1956,1999))
lines(predict(BEER.hw,n.ahead=48))
```

Figure 6.6 (bottom) shows the result.

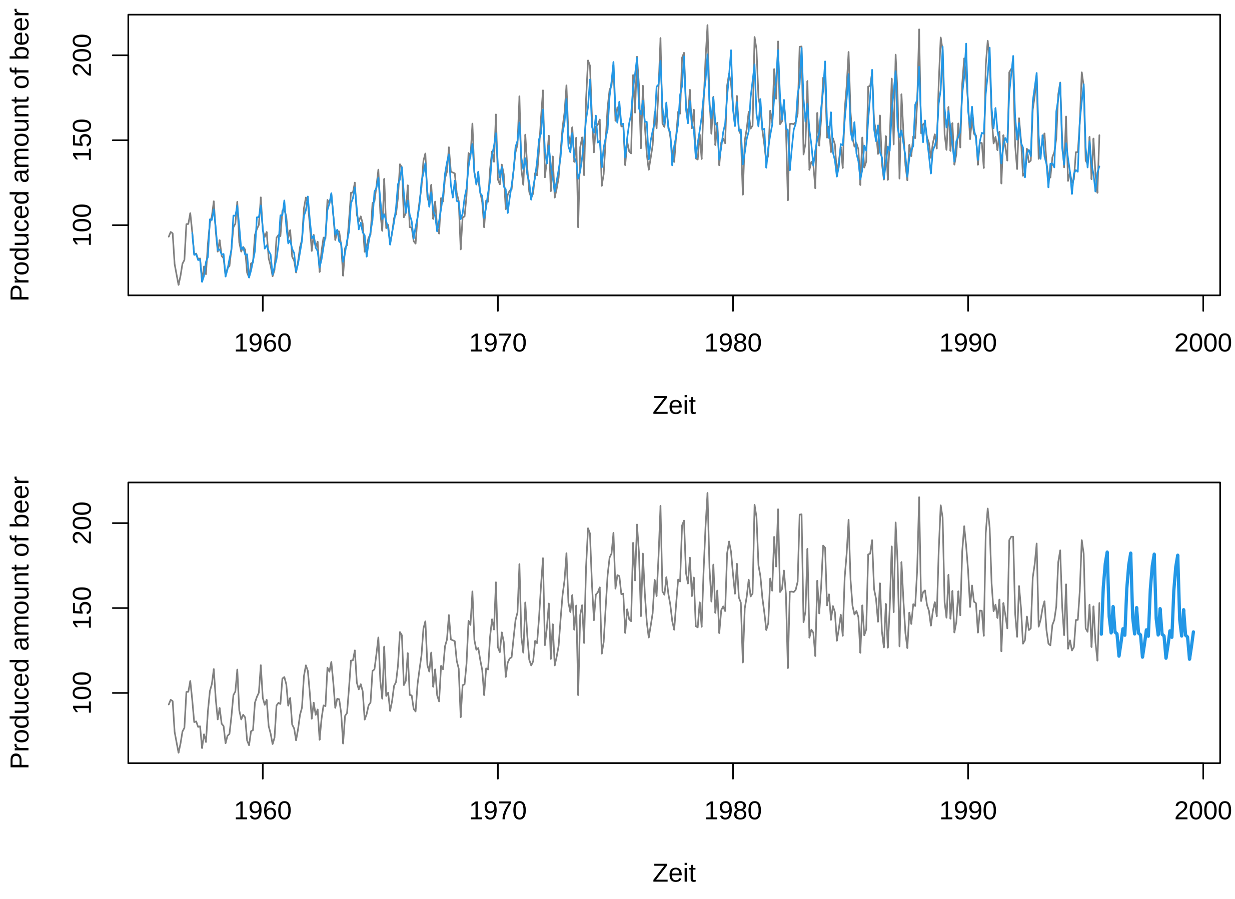


Figure 6.6: Exponential smoothing after Holt-Winters (top), and forecast (bottom).

6.4 Modelling time series

6.4.1 Important characteristics

Now we come to some prominent time series models, and they will make use of dependencies which are given by a systematic shift of the series by k steps (time points). Thus we consider the values x_t and x_{t-k}, where t is varying. In this context we talk about a *lag* of k.

Dependencies are analyzed with the **autocovariance**. The autocovariance of *order* k is defined as $\mathrm{Cov}(x_t, x_{t-k})$, and it can be estimated by

$$c_k = \frac{1}{T}\sum_{t=k+1}^{T}(x_t - \bar{x})(x_{t-k} - \bar{x}),$$

with the arithmetic mean $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$. The autocovariance for *lag* 0, thus c_0, is the variance of x_t. With this we can define the **autocorrelation of order** k as

$$\rho_k = \mathrm{Corr}(x_t, x_{t-k}) = \mathrm{Cov}(x_t, x_{t-k})/\mathrm{Var}(x_t),$$

and estimate it by $r_k = c_k/c_0$.

Above we have mentioned the terminology *stationary time series*. Those are characterized by the same expectation (mean) and by the same variance for all time points t, and their autocovariance is identical for all $k > 0$ and independent of time. Stationary time series with autocorrelation zero for $k > 0$ are named as *white noise*.

We can also test for the property of *white noise* by using the Q-statistic, also called **Ljung-Box statistic**. Here we test the hypothesis $H_0 : \rho_1 = \rho_2 = \ldots = \rho_{kmax} = 0$, with a fixed value of $kmax$.

Figure 6.7 shows the water level (annual mean, in feet) of lake Huron in the period 1875-1972. It is clear that there must be a time dependency, and this can also be seen in the plot. However, it is not at all clear how long this time dependency lasts. Figure 6.8 (left) shows

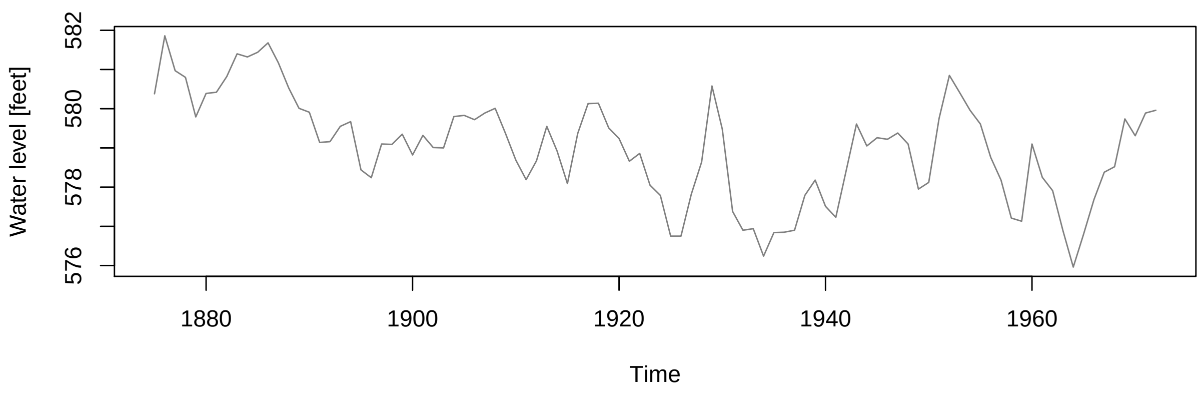


Figure 6.7: Mean annual water level of lake Huron.

autocorrelations for several values of k. The value k is called **lag**, and the plot is called **autocorrelation function (ACF)**. The dashed horizontal lines are the thresholds for significance, and they are determined by uncorrelated time series. For uncorrelated stationary time series, all autocorrelations for $k > 0$ should fall inside these boundaries. In this example we can see high autocorrelations and a slow decay over the lags; significance is for up to 9 lags.

Correlations between neighboring time points can be transferred, i.e. from x_t to x_{t-1}, from x_{t-1} to x_{t-2}, etc. The resulting correlation between x_t and x_{t-k} is thus influenced by the observations in between these time points. In order to eliminate this effect, there is another important characteristic for temporal dependency, called **partial autocorrelation** of order k:

$$\mathrm{Corr}(x_t, x_{t-k} | x_{t-1}, \ldots, x_{t-k+1}) \quad \text{for} \quad k = 0, 1, 2, \ldots$$

This corresponds to the autocorrelation between the residuals of a regression of x_t on $x_{t-1}, \ldots, x_{t-k+1}$, and the residuals of a regression of x_{t-k} on the same variables $x_{t-1}, \ldots, x_{t-k+1}$. Figure 6.8 (right) shows the partial autocorrelation function (PACF) of the lake Huron data. We can see significant partial autocorrelations up to lag 2.

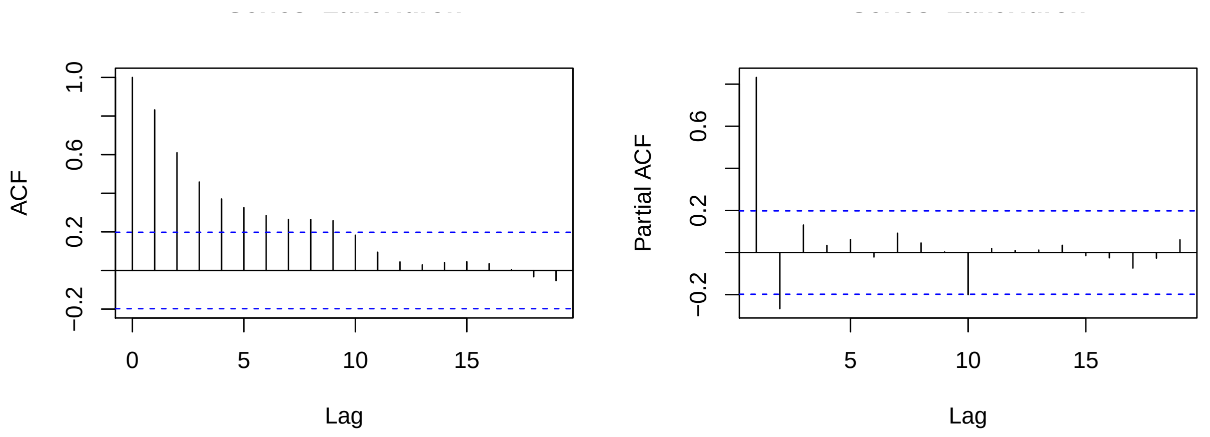


Figure 6.8: Autocorrelation function (left) and partial autocorrelation function (right) of the lake Huron data.

6.4.2 Basic time series models

Both the ACF and the PACF defined above are important for selecting an appropriate model for time series. Below we list some of the basic models which are already useful for many applications. However, there exists a multitude of different models which are designed for specific application, such as e.g. for financial time series which show a high volatility. Those models can be found in the literature.

Moving Average (MA) model

A *stationary* time series follows a *Moving Average* process of order 1, denoted as MA(1), if

$$x_t = a + u_t - \theta u_{t-1},$$

with the unknown parameters a and θ. The variables u_t and u_{t-1} represent *white noise* terms. Similarly, we can define a *Moving Average* process of order q, thus MA(q), as

$$x_t = a + u_t - \theta_1 u_{t-1} - \theta_2 u_{t-2} - \ldots - \theta_q u_{t-q},$$

with the unknown parameters a and $\theta_1, \ldots, \theta_q$.

The autocorrelations of MA(q) are 0 for lags bigger than q. An MA(q) process is characterized by an ACF plot where the autocorrelations show a strong decay and are no longer significant after lag q. The partial autocorrelations typically show a slow decay, with alternating sign. An example of a simulated MA(2) process is shown in Figure 6.9 (top).

Autoregressive (AR) model

A *stationary* time series follows an *autoregressive* process of order 1, denoted as AR(1), if

$$x_t = a + \phi x_{t-1} + u_t,$$

with the unknown parameters a and ϕ. Similarly, we can define an *autoregressive* process of order p, thus AR(p), as

$$x_t = a + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + u_t,$$

with the unknown parameters a and $\phi_1, \ldots, \phi_p$.

The partial autocorrelations of AR(p) are 0 for lags bigger than p. The autocorrelations show a slower decay towards 0, and sometimes their structure corresponds to a sine form. An example of a simulated AR(2) process is shown in Figure 6.9 (bottom).

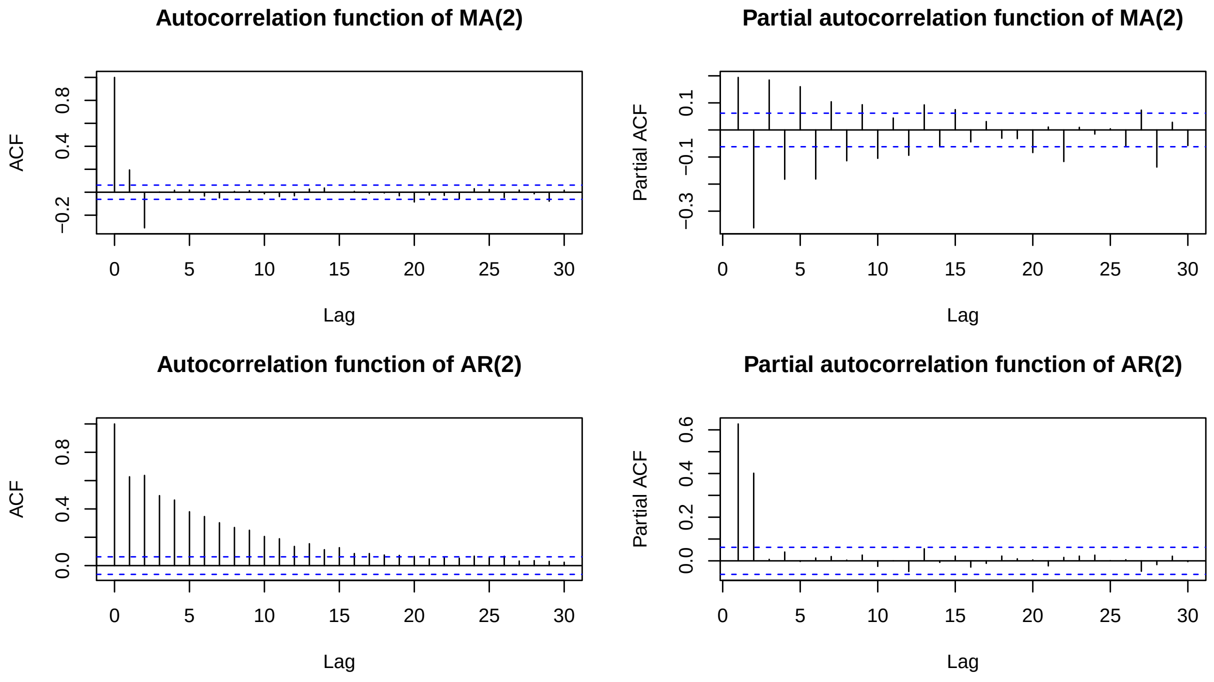


Figure 6.9: Typical structure of MA(2) and AR(2) processes, simulated with `arima.sim()`.

ARMA (Autoregressive-Moving-Average) model

If neither the ACF plot nor the PACF plot show an "implosion" to zero from a certain lag on, the process could be a mixture of AR(p) and MA(q), which is denoted as ARMA(p, q) and defined as

$$x_t = a + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + u_t - \theta_1 u_{t-1} - \theta_2 u_{t-2} - \ldots - \theta_q u_{t-q}.$$

ARIMA model

ARMA models require a stationary time series. Time series with a trend are named as non-stationary time series, or also as *integrated*. Trends can be eliminated by using *differences*. For this we define the *difference operator* Δ as

$$\Delta x_t = x_t - x_{t-1}.$$

Linear trends can be eliminated by applying Δ once, while quadratic trends can be eliminated by applying Δ twice,

$$\Delta^2 x_t := \Delta(\Delta x_t) = \Delta(x_t - x_{t-1}) = x_t - 2x_{t-1} + x_{t-2}.$$

Definition: A time series x_t, $t = 1, \ldots, T$, follows an ARIMA(p, d, q) model if $\Delta^d x_t$ follows an ARMA(p, q) model.

Generally, simple models with only few parameters are to be preferred, and therefore d is usually chosen as 0 or 1.

Example: An ARIMA(1,1,0) model is of the form

$$\Delta x_t = a + \phi \Delta x_{t-1} + u_t,$$

and thus

$$x_t = x_{t-1} + a + \phi(x_{t-1} - x_{t-2}) + u_t.$$

6.4.3 Parameter estimation

Parameter estimation for the above models is done according to the least-squares principle, thus by minimizing $\sum_{t=1}^{T}(x_t - \hat{x}_t)^2$. Here we do not go into more technical details with the parameter estimation. In `R` we can estimate the parameters by

```
arima(data,order=c(p,d,q))
```

for the time series `data`, with the corresponding orders of the ARIMA(p, d, q) model.

For the lake Huron time series we look at Figure 6.8 and try the following model:

```
data(LakeHuron)
fit <- arima(LakeHuron,order=c(1,0,1))
```

This is an ARMA(1,1) model of the form

$x_t = a + \phi x_{t-1} + u_t - \theta u_{t-1}.$

The estimations of the parameters ϕ, θ and a are:

```
> fit$coef
       ar1         ma1   intercept
  0.7448993   0.3205891 579.0554556
```

For identifying an appropriate model, the function `auto.arima()` from the package `fore-cast` could be helpful. This function fits ARIMA models with various different parameter combinations, and compares those models based on model selection criteria.

6.4.4 Diagnostic tools

So far we have seen that selecting an appropriate model could be a bit tricky. We shall select not only the appropriate underlying process (MA, AR, ARMA, ARIMA), but also the order of the model, and thus the number of parameters to be estimated. Generally, the order p and q should be chosen "small" in order to avoid overfit to the existing data, which might be important for good predictive power.

Thus, a crucial step for model selection is diagnostics. For the lake Huron data above we have chosen an ARMA(1,1) model, and `R` gives back the estimated parameters. However, we have no idea whether the model is appropriate. Diagnostic plots can be generated in `R` by:

```
tsdiag(fit)
```

The result is shown in Figure 6.10. The plot on top presents (standardized) residuals, which should be white noise if the model is correctly specified. The middle plot is the ACF plot of these residuals, and it should not contain any significant correlations (except that at lag 0). The bottom plot presents the Ljung-Box statistic with p-values up to lag 10; these should be above 0.05, thus non-significant. Overall, our model seems to be well specified.

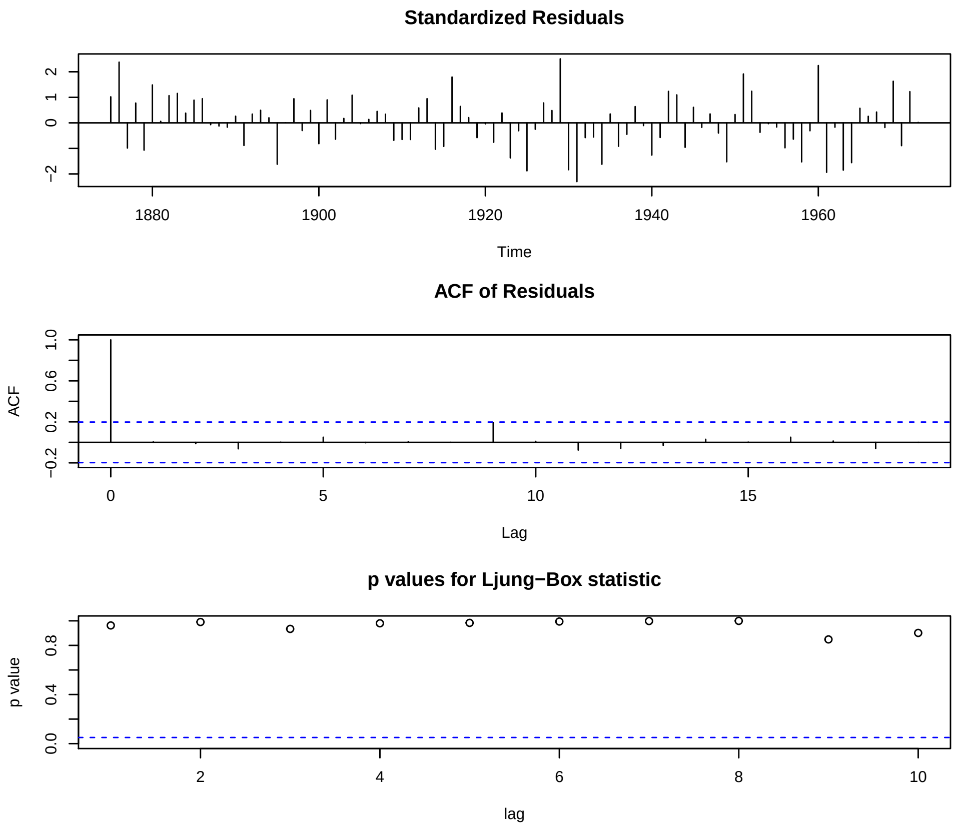


Figure 6.10: Diagnostic plots for the ARMA(1,1) model applied to the lake Huron time series.

6.4.5 Forecasting time series

Since now we have specified a meaningful model, we can come to the most interesting part: to the forecast. Note that a meaningful model is not necessarily a good prediction model. Our result object `fit` also contains information about the variance of the estimates, which can be used to construct confidence intervals in addition to the predictions.

```
plot(LakeHuron,xlim=c(1875,1980))
LH.pred <- predict(fit,n.ahead=8)  # forecast for the next 8 years
lines(LH.pred$pred,col="blue")
lines(LH.pred$pred+2*LH.pred$se,col="blue",lty=2)
lines(LH.pred$pred-2*LH.pred$se,col="blue",lty=2)
```

Figure 6.11 shows the forecast for the next 8 years, jointly with a 95% confidence interval. The width of this interval reveals that – although the model seems to make sense – it is not really valuable for prediction.

At the web page

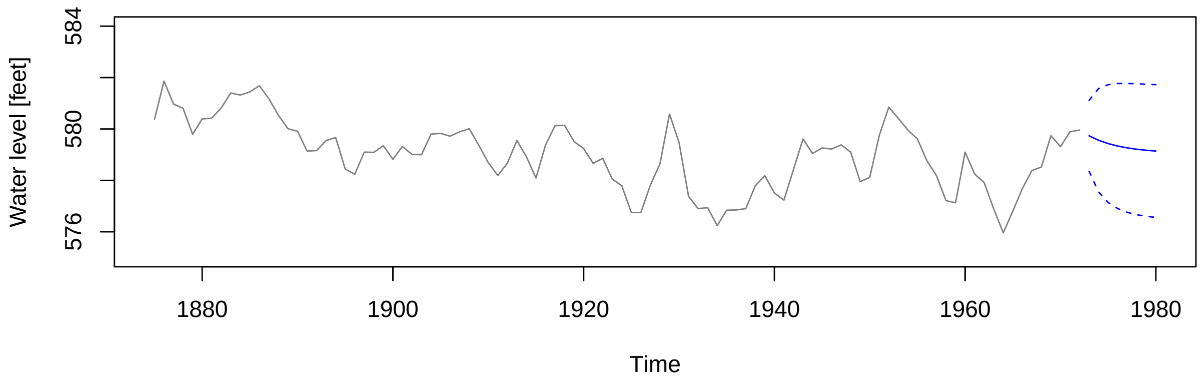


Figure 6.11: Forecast of the water level of lake Huron for the next 8 years, with confidence intervals.

```
http://www.glerl.noaa.gov/data/now/wlevels/lowlevels/
        plot/data/Michigan-Huron-1860-.csv
```

one can find historical and more recent data of the annual mean water level of lake Huron. Figure 6.12 shows these data similar as in Figure 6.11, but with an additional red line. The data from `R` do not exactly match these new data – probably the measurements have been taken from another location at the lake. In this plot we have done a forecast with our model up to the year 2011, which allows to compare with the newly imported data. The confidence interval covers essentially the whole data range, and the forecast quickly stabilizes to the mean. The forecast is – if at all – only useful for a very short period of time. Our model either seems to be too simple, or one can simply not find a better model.

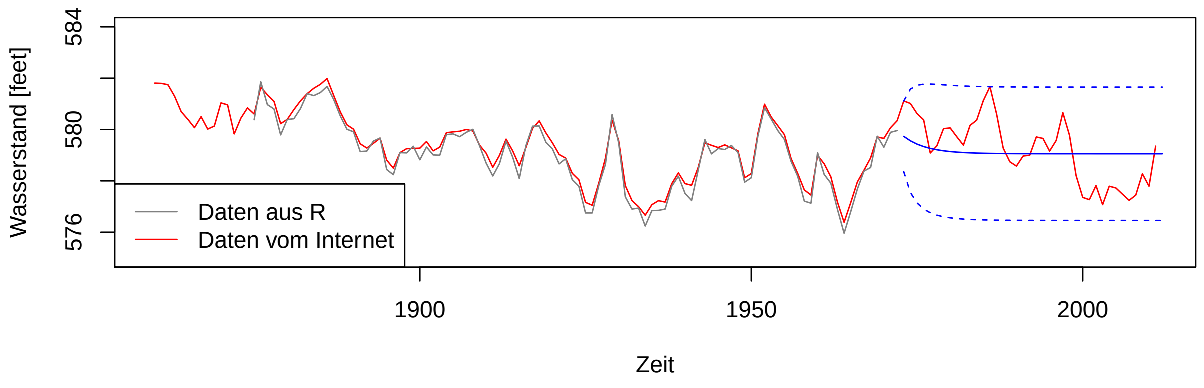


Figure 6.12: Comparison of the forecast with the newly imported data from lake Huron.

Chapter 7

Multivariate graphics

Multivariate data are usually presented in a rectangular table (matrix), consisting of n rows for the observations, and of p columns for the variables. In the following this matrix will be denoted by $\boldsymbol{X}$. The rows (samples, observations) are denoted by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The i-th sample is thus $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ (column vector!).

7.1 Scatter diagrams

Scatter diagrams (or scatter plots) for low-dimensional data could still be used to present the multivariate data information, by varying symbols, symbol sizes, symbol color, etc. As an example we show the well known Iris data (`data(iris)`) which contain measurements of 4 characteristics of three different species of the Iris flower. The first two dimensions *Sepal Length* and *Sepal Width* are used for the horizontal and vertical axis, respectively. The variable *Petal Length* is represented by the symbol size, and variable *Petal Width* by different gray scale of the symbols. One can quite well see that the points fall into 2 clusters, however, since the observations originate from 3 species, they should rather form 3 clusters, which is not visible in this plot.

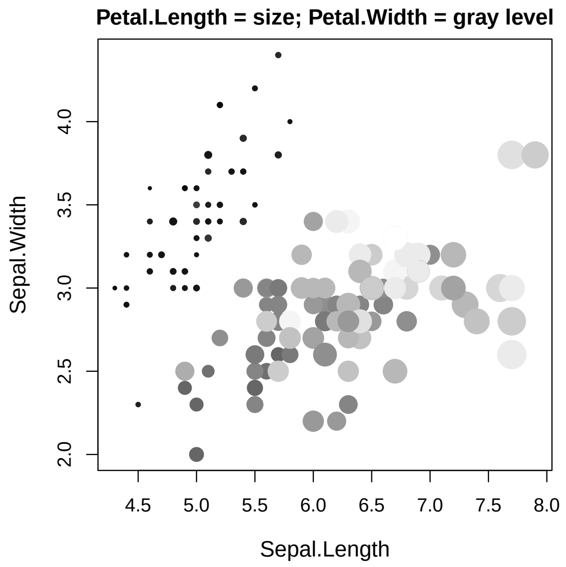


Figure 7.1: Iris data: the four variables are represented in the plot by encoding two of them in symbol size and color.

Another possibility is to present all $\binom{p}{2}$ two-dimensional plots of the p variables in a *scatterplot matrix*, see Figure 7.2. Here we also present the species information by different colors. Indeed, two of the groups are hard to distinguish.

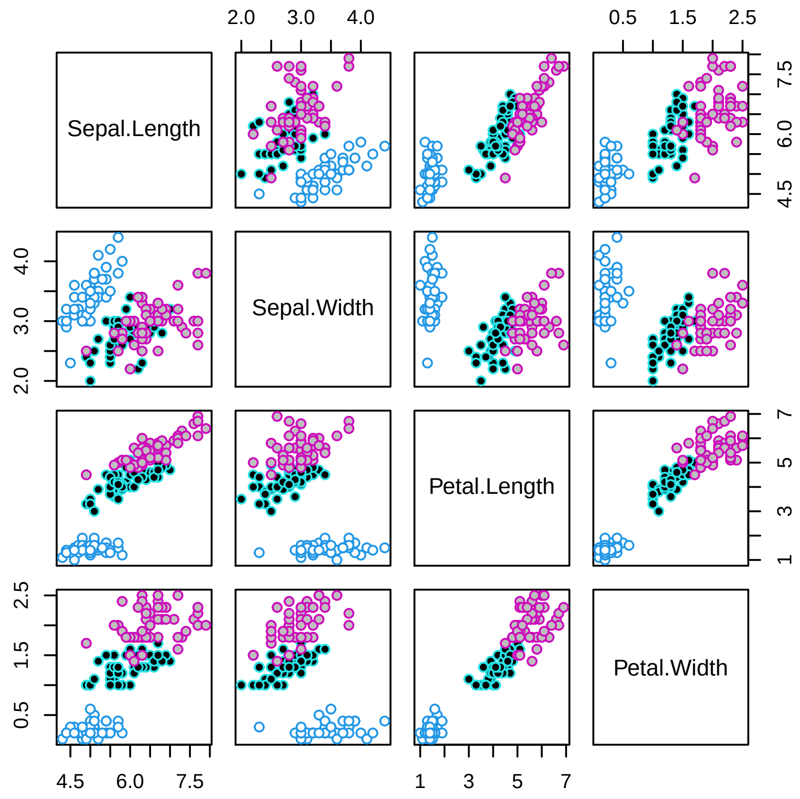


Figure 7.2: Iris data: scatterplot matrix, with collors for the different Iris species.

Note that this representation of all pairs of variables only shows very specific two-dimensional plots, namely those which are "projected" to the axes representing the variables. In principle one could use any projection directions, and those plots could be better suited to reveal group structures or patterns.

7.2 Profiles, stars, segments, Chernoff faces

7.2.1 Profiles

Every observation $\boldsymbol{x}_i$ should be represented by p bars, where the length of the j-th bar should be proportional to x_{ij}.

As an example we consider the data set from Table 7.1 with death rates in Virginia. The "variables" are the different population groups (columns), and the "observations" are the age groups (rows). Figure 7.3 shows profiles of the different observations (left). However, we could also transpose this table, and then the observations (profiles) would be the population groups. This is shown in the right plot.

The example we have presented is a very specific type of a profile plot. Usually we have data measured for different variables, possibly with very different scale. Then it would be

Table 7.1: Death rates (in %) in Virginia in the year 1940. The data are split into age groups (rows) and population groups (columns).

Age group	Rural Male	Rural Female	Urban Male	Urban Female
50-54	11.7	8.7	15.4	8.4
55-59	18.1	11.7	24.3	13.6
60-64	26.9	20.3	37.0	19.3
65-69	41.0	30.9	54.6	35.1
70-74	66.0	54.3	71.1	50.0

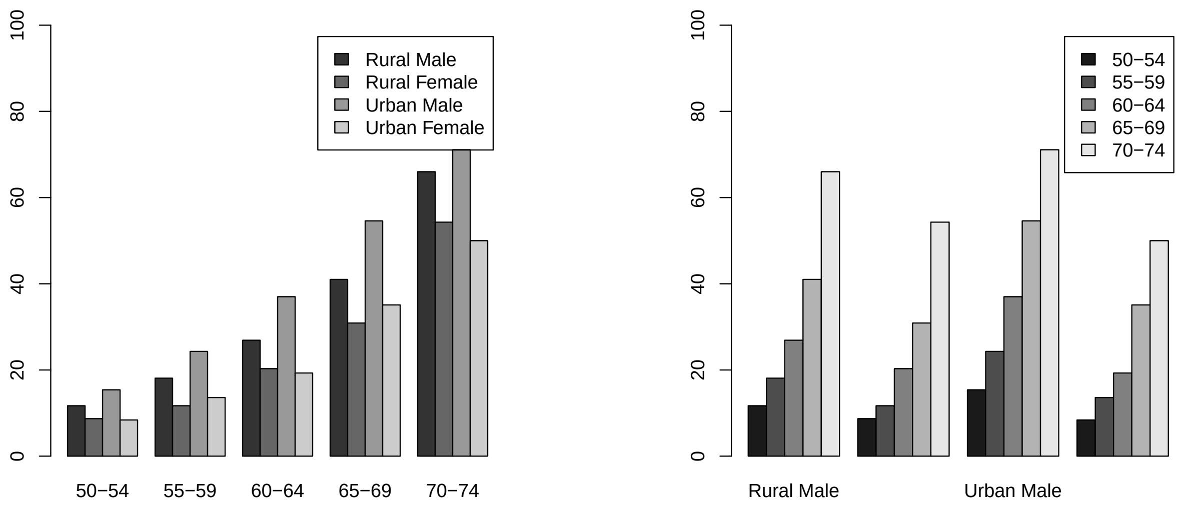


Figure 7.3: Profile plots of the data from Table 7.1, with profiles for the original matrix (left) and the transposed matrix (right).

necessary to first make the information comparable. A common approach in multivariate graphics is to perform a $[0, 1]$ scaling of every variable, thus to visualize the values

$$x^*_{ij} = \frac{x_{ij} - \min_i(x_{ij})}{\max_i(x_{ij}) - \min_i(x_{ij})} \qquad \text{for} \qquad j = 1, \ldots, p.$$

An alternative way of scaling would be a transformation of the variables to mean zero and variance one.

7.2.2 Stars

We construct for every observation a star-type symbol, by arranging the p variable axes radially in equal angles $\frac{2\pi}{p}k$ $(k = 0, \ldots, p-1)$. Then the (scaled) values of the observations are drawn as distances from the origin, and those resulting points are connected by lines.

Figure 7.4 shows star plots for the data set `data(mtcars)`, containing information of 32 different cars for 7 car characteristics, as reported in a US motor journal from the year 1974. Note that according to the $[0, 1]$ scaling, every variable will (at least) once have a value of zero, and thus the stars have a very specific form. Still, one can compare the star shapes and probably visually identify groups of similar cars.

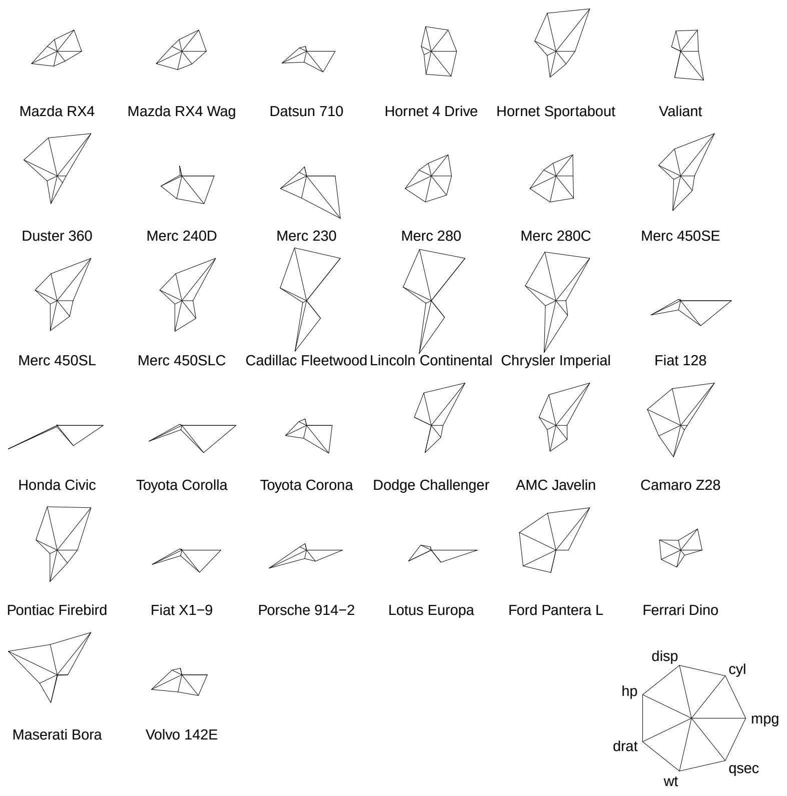


Figure 7.4: Representation of the car data set by star symbols. (`stars`)

7.2.3 Segments

Segments are very similar to stars. Again, the circle is split into regular angles, but this time the area of each segment represents the data value. Figure 7.5 presents our car data. Here also different gray scales are used, which, however, could already introduce some subjectivity (colors even more).

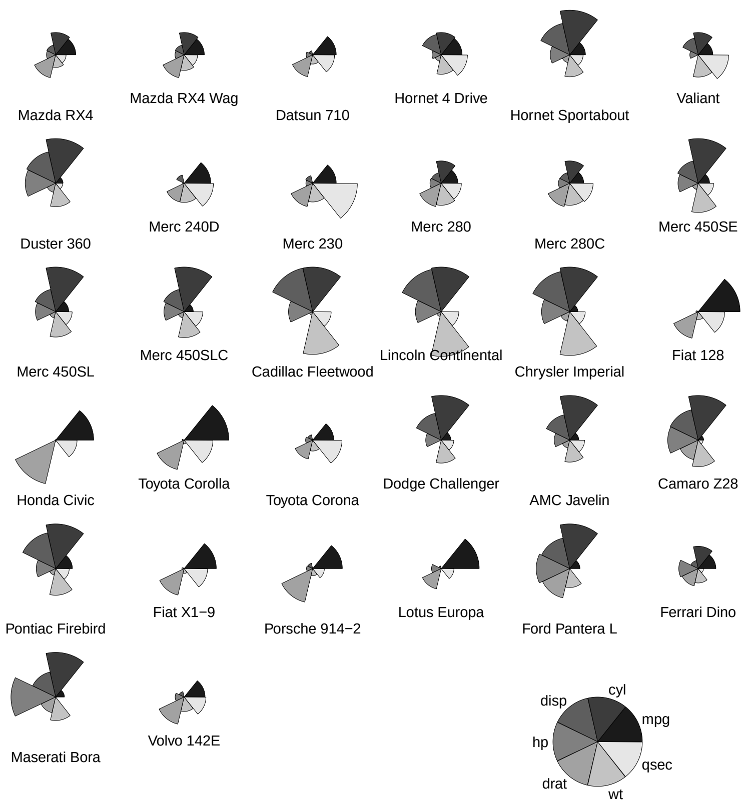


Figure 7.5: Representation of the car data by segments. (`stars`)

7.2.4 Chernoff faces

The idea is to construct "faces", and the different variables are encoded into the different characteristics of a face, such as shape and color of face, size of nose and eyes, distance between eyes, etc. As one can imagine, the visual impression could be biased by the variables which are encoded into the "most important" face characteristics, and thus this plot type could result in subjectivity. Figure 7.6 shows Chernoff faces for the cars data set.

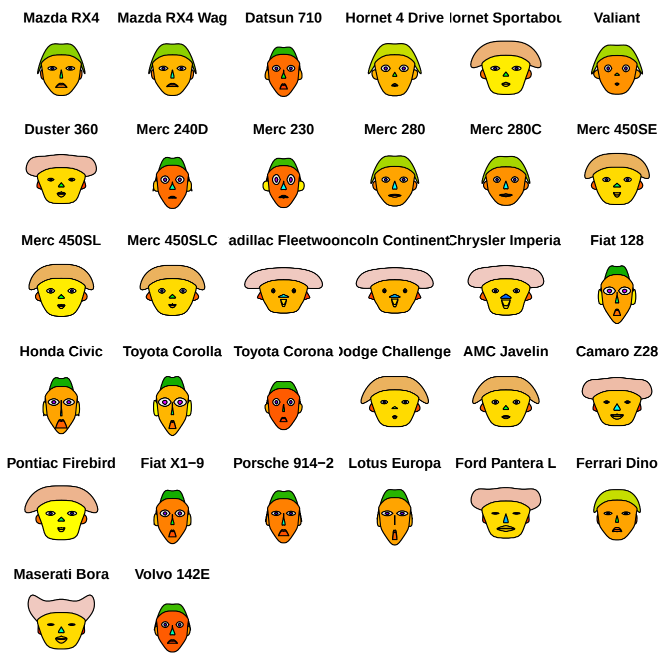


Figure 7.6: Representation of the cars data by Chernoff faces. (`faces` from `library(aplpack)`)

7.2.5 Boxes

Here the p variables are first grouped into 3 classes. This is based on the similarity between the variables, which could be expressed by the correlations. Then every variable group represents one direction of a box. The size of the box is determined by the relative proportion of the observations in the groups. Figure 7.7 shows boxes for the cars data set.

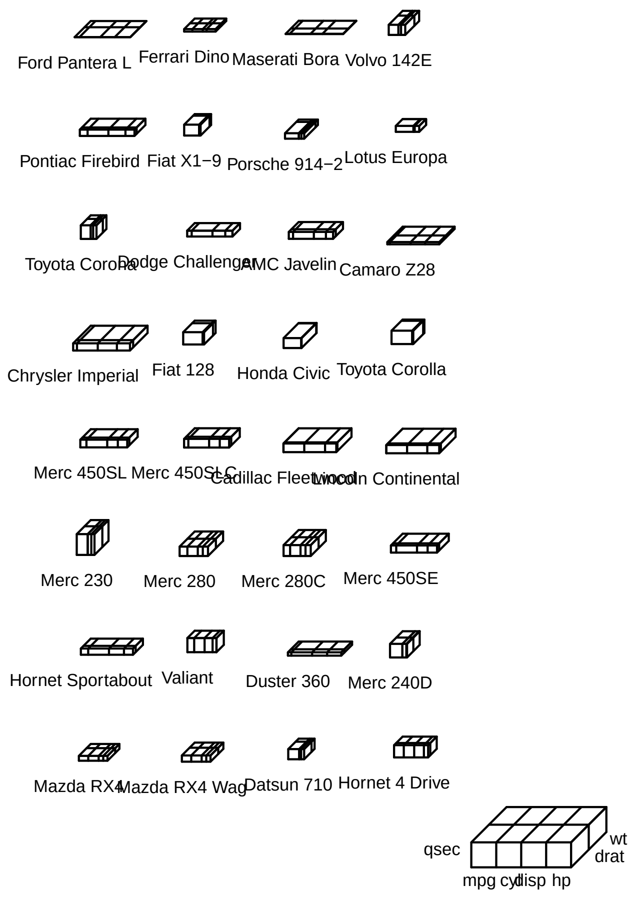


Figure 7.7: Representation of the cars data set by boxes. (`boxes` aus `library(StatDA)`)

7.3 Trees

Trees try to express the correlation structure between the variables, which is represented by the sequence of the branches of the trees. The variables or variable groups form branches according to their similarity. The width of a branch is proportional to the number of variables, and also the angle to the vertical direction is proportional to this number. The lengths of the branches are proportional to the average of the values of an observation for these variables.

Figure 7.8 shows trees for the car data set. The symbol legend shows very clearly the hierarchical correlation structure between the variables.

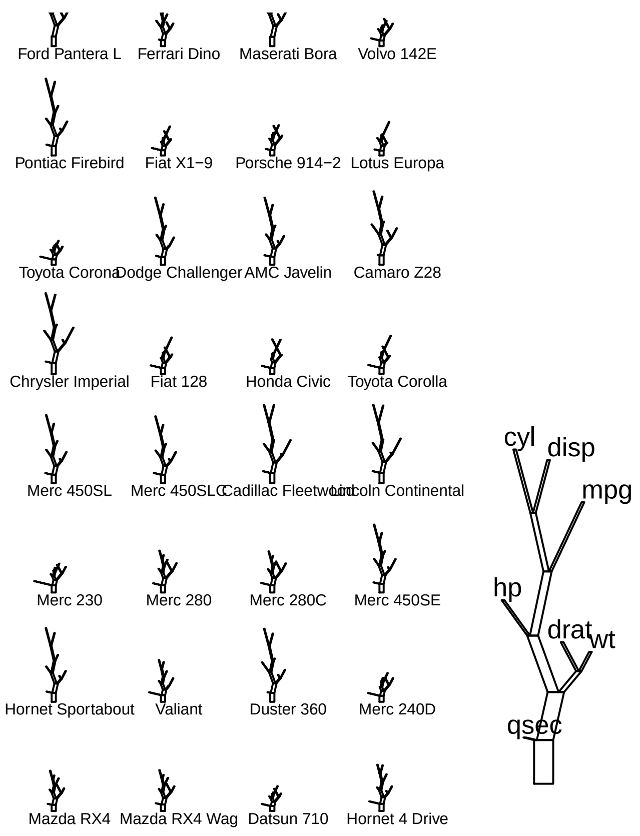


Figure 7.8: Car data set represented by trees.

7.4 Castles

Castles can be viewed as very specific trees, where the “branches” are drawn with an angle of zero to the vertical direction. Fiigure 7.9 shows again the car data set with castles.

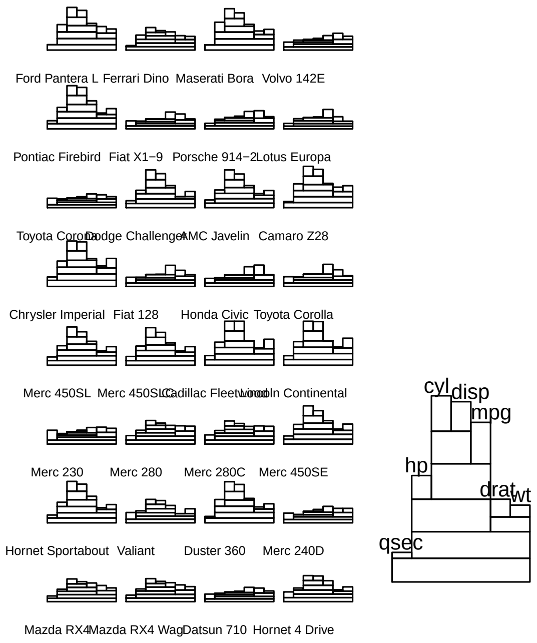


Figure 7.9: Representation of the car data by castles.

7.5 Parallel coordinates

This type of visualization is quite different to the previous ones. We do not construct a special symbol for every observation. Rather, an observation is represented as a line, passing parallel coordinates. These coordinates are the p variables, with values scaled to $[0, 1]$, and the coordinate values are simply the (rescaled) values of the observations for these variables.

Figure 7.10 shows another car data set, available as data frame `Auto` from the `library(ISLR)`. The last variable “origin” represents the origin American (light blue), European (pink) and Japanese (yellow). One can quite clearly see that most of the American cars have very different characteristics. We can also see that categorical variables (e.g. the number of cylinders) suggest somehow a clustering structure which might not be present. Moreover, a reordering of the variables could improve the visual impression and make data subgroups better visible.

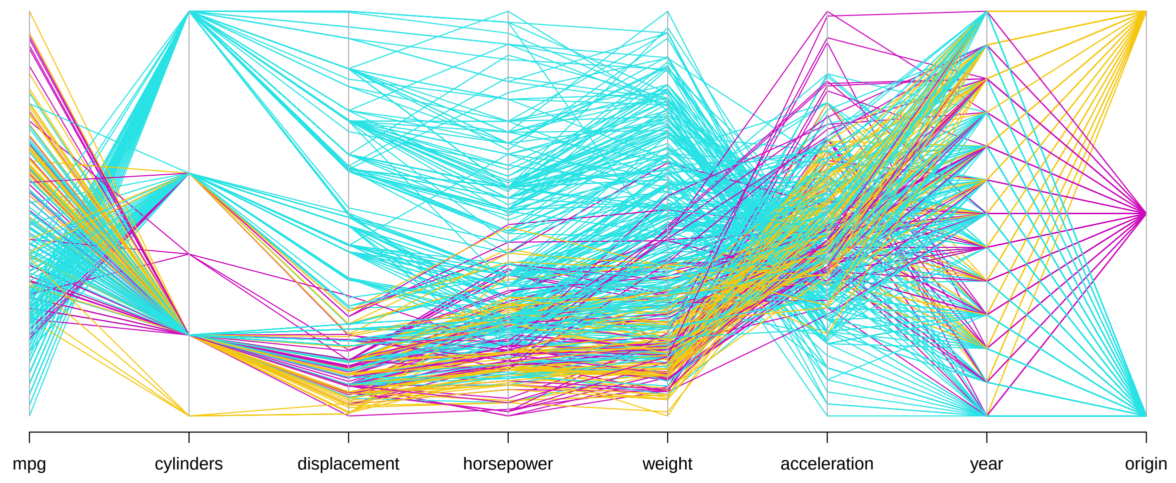


Figure 7.10: Representation of car characteristics with parallel coordinates. (`parcoord`)

Chapter 8

Multivariate parameter estimation

Chapter 2 has already presented several different estimators of location and scale in the univariate case. Here we want to move on to the multivariate case. We assume to have n observations measured for p variables $x_1, \ldots, x_p$. We could again use the univariate estimators of location and scale separately for each variable, but now we are also interested in the relationships between the variables.

8.1 Covariance and correlation

Covariance and correlation are possibilities to characterize the relationship between the variables. Theoretically, the **covariance** between a pair of random variables x_j and x_k ($j, k \in \{1, \ldots, p\}$) is defined as

$$\sigma_{jk} = \mathrm{E}[(x_j - \mathrm{E}(x_j))(x_k - \mathrm{E}(x_k))]$$

where the symbol "E" denotes the expectation. For $j = k$ we obtain the theoretical **variance**, here denoted as σ_{jj}, for $j = 1, \ldots, p$.

The population **correlation coefficient** between x_j and x_k is defined as

$$\rho_{jk} = \frac{\sigma_{jk}}{\sqrt{\sigma_{jj}\sigma_{kk}}},$$

and it is a dimension-free measure for the linear relationship between these two random variables, which is always in the interval $[-1, 1]$. Therefore, the correlation is easier to interpret than the covariance because it does not depend on the variances of the variables. If the coefficient is equal to 1 or -1, the variables x_j and x_k contain the same information, but for -1 there is a reverse relationship. If the coefficient is 0, we have no linear relationship between x_j and x_k.

If we have collected n observations simultaneously for the p variables in an $n \times p$ data matrix $\boldsymbol{X}$, we can estimate covariance and correlation. Consider variable x_j with the measurements $x_{1j}, \ldots, x_{nj}$, and variable x_k with the values $x_{1k}, \ldots, x_{nk}$. The classical estimator for the covariance σ_{jk} is the **sample covariance**

$$s_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k),$$

with the arithmetic means

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \quad \text{and} \quad \bar{x}_k = \frac{1}{n}\sum_{i=1}^{n} x_{ik}.$$

For $j = k$ we obtain the **sample variance**, see Section 2.1.

The classical estimator for the correlation coefficient ρ_{jk} is the **sample correlation coefficient**, also called *Pearson correlation*, which is defined as

$$r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj}s_{kk}}}.$$

Again, if $r_{jk} = 0$, then there exists no *linear* relationship between x_j and x_k; however, there could still exist a nonlinear relationship.

If we determine the covariance (or correlation) for all pairs of variables, we obtain a matrix of dimension $p \times p$, the so-called covariance (or correlation) matrix. The theoretical **covariance matrix** with all elements σ_{jk} is denoted by $\boldsymbol{\Sigma}$. The sample covariance matrix with the elements s_{jk} is denoted by $\boldsymbol{S}$. The theoretical correlation matrix is denoted by $\boldsymbol{\rho}$, and its sample counterpart by $\boldsymbol{R}$.

In R we can obtain the sample covariance matrix and the sample correlation matrix for a data matrix $\boldsymbol{X}$ by

```
R:  S <- cov(X)    # sample covariance matrix
R:  R <- cor(X)    # sample correlation matrix
```

Figure 8.1 should give some impression about the correlations based on simulated data. In each of the plots we can see 200 observations, and they are generated from a bivariate normal distribution with correlation 0.8 (left), 0 (middle), and −0.8 (right). The sample correlation coefficients will very likely be close to these theoretical values.

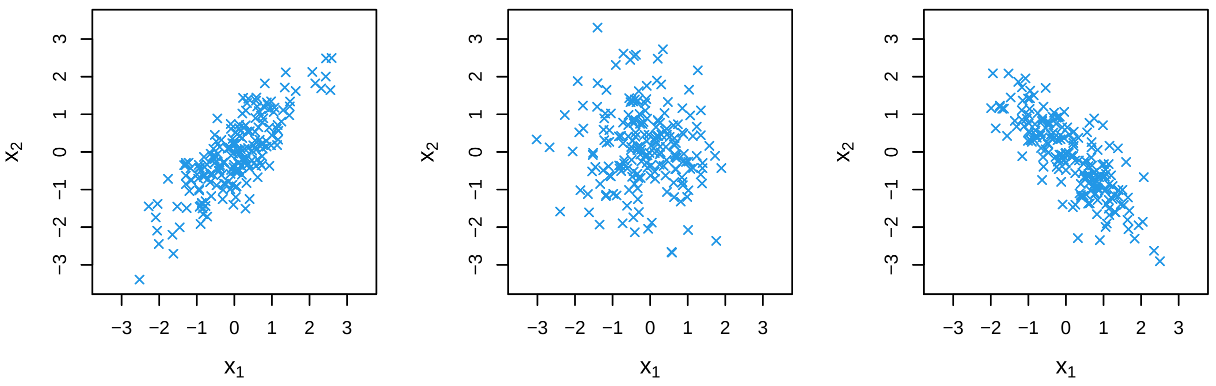


Figure 8.1: Data points generated from bivariate normal distributions with correlation 0.8 (left), 0 (middle), and −0.8 (right).

8.1.1 Robust estimation of covariance and correlation

As already noted before, both arithmetic mean and sample variance are very sensitive to data outliers. More robust estimators are median and MAD (and we have seen several more estimators). Similarly, also the sample covariance is sensitive to outliers because values far away could heavily determine the resulting estimate.

More robust estimators of covariance or correlation would have to downweight observations which clearly deviate from the "data majority". Note, however, that a "robustified" pairwise estimation of the elements of the covariance or correlation matrix might be problematic: depending on the pair, one would probably have to downweight different observations, and this could lead to an "inappropriate" robustly estimated matrix.

One approach for a more robust estimation is the **Spearman rank correlation**: here we compute the Pearson correlations from the ranks of the observations, which are values between 1 (minimum) and n (maximum of all values of a variable). Since we assign ranks, the range is automatically limited, and also nonlinearities could be covered by this measure. In other words, Spearman's rank correlation is a measure of *monotone relationship*.

```
R:  cor(X,method="spearman")
```

A more robust estimator of covariance is the **MCD** (*Minimum Covariance Determinant*) **estimator**. The idea is related to the LTS estimator of regression, see Section 4.6: We consider a subsample of size h of the observations, where h is between $n/2$ and n. For this subsample we compute the classical sample covariance matrix (for LTS regression we computed the classical LS fit). The MCD estimator is then defined by that subsample of size h which has the smallest determinant of its sample covariance matrix. This covariance matrix still needs to be multiplied by a constant for consistency. As a by-product we can also obtain a robust multivariate location estimator, simply by taking the arithmetic mean of the final subsample.

```
R:  library(robustbase)
R:  covMcd(X)
```

It is now immediate how to obtain a robust estimator for the correlation matrix. Denote by $\boldsymbol{C}$ the MCD estimator for the covariance matrix, and its elements by c_{jk}, for $j, k = 1, \dots, p$. The robust variances are given by c_{jj}. The robust correlation matrix is given according to the definition of a correlation, namely by

$$\frac{c_{jk}}{\sqrt{c_{jj}}\sqrt{c_{kk}}}$$

for all j and k.

8.2 Distance and similarity

While we have discussed the relationships between the variables in the previous section, we will focus now on the relationships between the observations. Interestingly, both kinds of relationships are based on the same data! Let us again consider observations in the p-dimensional space, in particular the two observations $\boldsymbol{x}_A = (x_{A1}, \dots, x_{Ap})^T$ and $\boldsymbol{x}_B = (x_{B1}, \dots, x_{Bp})^T$. These are also shown in Figure 8.2 in two dimensions. We can now define different distance measures for these two points:

The **Euclidean distance** between $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ is defined as

$$d_E(\boldsymbol{x}_A, \boldsymbol{x}_B) = \left(\sum_{j=1}^{p} (x_{Bj} - x_{Aj})^2 \right)^{1/2} = [(\boldsymbol{x}_B - \boldsymbol{x}_A)^T (\boldsymbol{x}_B - \boldsymbol{x}_A)]^{1/2} = \|\boldsymbol{x}_B - \boldsymbol{x}_A\|.$$

We call $\|\cdot\|$ the Euclidean norm.

```
R:  dist(X,method="euclidean")
```

The **Manhattan distance** (also called *city block* distance) is based on the absolute coordinate-wise distances:

$$d_M(\boldsymbol{x}_A, \boldsymbol{x}_B) = \sum_{j=1}^{p} |x_{Bj} - x_{Aj}|$$

```
R:   dist(X,method="manhattan")
```

The above distance measures can be generalized to the **Minkowski distance**, defined as

$$d_{Mink}(\boldsymbol{x}_A, \boldsymbol{x}_B) = \left(\sum_{j=1}^{p} |x_{Bj} - x_{Aj}|^m \right)^{1/m},$$

with the parameter m as the (inverse) power.

```
R:   dist(X,method="minkowski",p=m)
```

A somewhat different concept is the **cosine of the angle** α between the observation vectors, which serves as a similarity measure between the observations. This measure is independent of the lengths of the vectors and thus only considers the relative values of the variables:

$$\cos\alpha = \frac{\boldsymbol{x}_A^T \boldsymbol{x}_B}{\sqrt{(\boldsymbol{x}_A^T \boldsymbol{x}_A)(\boldsymbol{x}_B^T \boldsymbol{x}_B)}} = \frac{\boldsymbol{x}_A^T \boldsymbol{x}_B}{\|\boldsymbol{x}_A\| \cdot \|\boldsymbol{x}_B\|}$$

Generally, a distance measure d can be converted into a similarity measure, e.g. by taking $1 - d/d_{max}$, where d_{max} is the maximum distance between all pairs of observations.

Figure 8.2 shows these different distance measures in two dimensions.

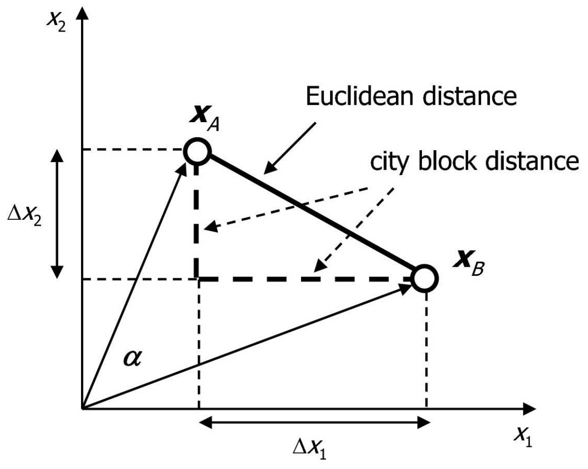


Figure 8.2: Different distance measures, visualized for two dimensions.

Another important distance measure is the **Mahalanobis distance** (after the Indian statistician Prasanta Chandra Mahalanobis). This distance measure accounts for the covariance structure of the data, and thus it does not depend on the scale of the variables. Usually, this distance is most interesting for an observation $\boldsymbol{x}_i$ to the center $\boldsymbol{\mu}$ of the distribution, where it is defined as

$$d_{Mahal}(\boldsymbol{x}_i, \boldsymbol{\mu}) = [(\boldsymbol{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu})]^{1/2} \qquad \text{for } i = 1, \ldots, n.$$

If we would replace the covariance matrix $\boldsymbol{\Sigma}$ by the identity matrix $\boldsymbol{I}$, we would obtain the Euclidean distance (of an observation to the center).

Figure 8.3 compares the Euclidean distance (left) with the Mahalanobis distance (right). Every point on a circle (ellipse) has the same distance to the center, and the distances increase for circles (ellipses) further away from the center. It is clear that the Mahalanobis distance is better suited than the Euclidean distance if the variables are correlated, e.g. with the covariance indicated in the right plot. We can also see that in case of uncorrelatedness (or $\boldsymbol{\Sigma} = \boldsymbol{I}$), both distance measures lead to the same answer.

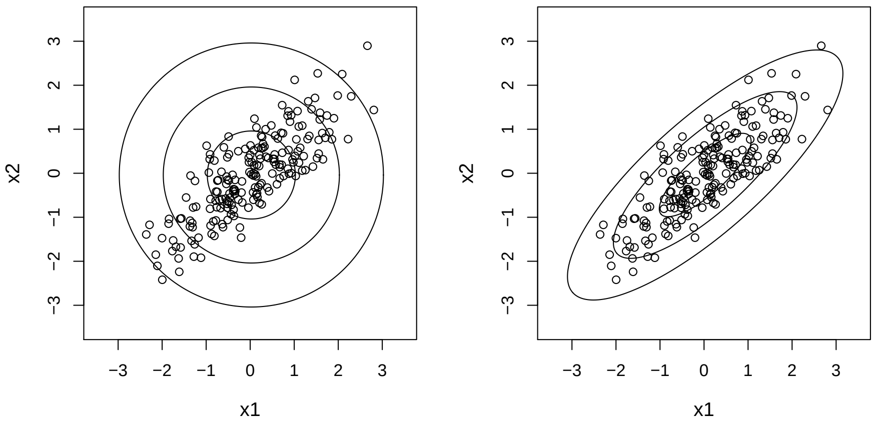


Figure 8.3: Euclidean distance (left) and Mahalanobis distance (right) to the center of the distribution. The circles (left) and ellipses (right) represent contour lines; every point there has the same distance to the center.

8.3 Multivariate outlier detection

The Mahalanobis distance is very useful for the porpose of identifying outliers in the multivariate data space. Outliers could be observations which are "far away" along one coordinate, but those observations could very likely be identified even by univariate outlier detection methods (boxplot, etc.). However, there can also be observations which are not extreme along a coordinate, but unusual concerning the joint data distribution. Such outliers cannot necessarily be identified by univariate methods.

In order to apply the Mahalanobis distance for multivariate outlier detection, we first have to estimate the location parameter $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. Since we would like to identify outliers, we need to consider robust estimators. One option is to use the MCD estimator, see Section 8.1.1, which yields a robust location estimator, say $\hat{\boldsymbol{\mu}} = \boldsymbol{t}$, and a robust covariance estimator, say $\hat{\boldsymbol{\Sigma}} = \boldsymbol{C}$. These can be plugged into our formula for the Mahalanobis distance,

$$d_{Mahal}(\boldsymbol{x}_i, \boldsymbol{t}) = [(\boldsymbol{x}_i - \boldsymbol{t})^T \boldsymbol{C}^{-1} (\boldsymbol{x}_i - \boldsymbol{t})]^{1/2} \qquad \text{for } i = 1, \ldots, n,$$

which results in a distance for every observation to the center with respect to the covariance. Observations with a "big" distance would be candidates for outliers. However, what means "big"? What we need is a threshold or outlier cutoff value to distinguish regular observations from multivariate outliers.

How did we obtain a threshold for univariate outlier detection? If we assume that the (majority of) the univariate data comes from a normal distribution, then $(x_i - \hat{\mu})/\hat{\sigma}$ follows (approximatively) a $N(0, 1)$, and the quantiles 0.025 and 0.975, thus the values ± 1.96 can be used as outlier thresholds. Now let us again assume that the data majority originates from a multivariate normal distribution, with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Then one can show that d^2_{Mahal} approximately follows a chi-square distribution with p degrees of freedom, thus χ^2_p. It is then common to use as an outlier cutoff value the quantile $\chi^2_{p;0.975}$. Values bigger than this threshold are considered as multivariate outliers.

Figure 8.4 presents an example for multivariate (bivariate) outlier detection. The data set `glass` from `library(chemometrics)` is used, consisting of measurements of 13 chemical elements (compounds) for 4 different types of glass. For simplicity we just consider MgO and Cl, and only 2 glass types, where 120 observations are from the first type, and 10 from the second. Thus, the 10 observations could have a different data structure and therefore represent multivariate outliers. Indeed, the scatterplot reveals that these observations (in pink) deviate somewhat from the data majority. The plots also show ellipses. These are so-called 97.5%-*tolerance ellipses*; this means that any point on this ellipse has a Mahalanobis distance equal to $\sqrt{\chi^2_{2;0.975}} = 2.72$, which is just the outlier cutoff value in this case. For the left plot, the classical arithmetic mean and sample covariance is used to compute the Mahalanobis distances, while for the right plot the MCD counterparts are taken, leading to robust distances. It can be seen that the outliers (observations from the second glass type) artificially inflate the ellipse based on the classical estimators, and they even lead to a covariance estimate with a different sign than the robustly estimated covariance. The corresponding robust distances are more reliable for the identification of the second glass type.

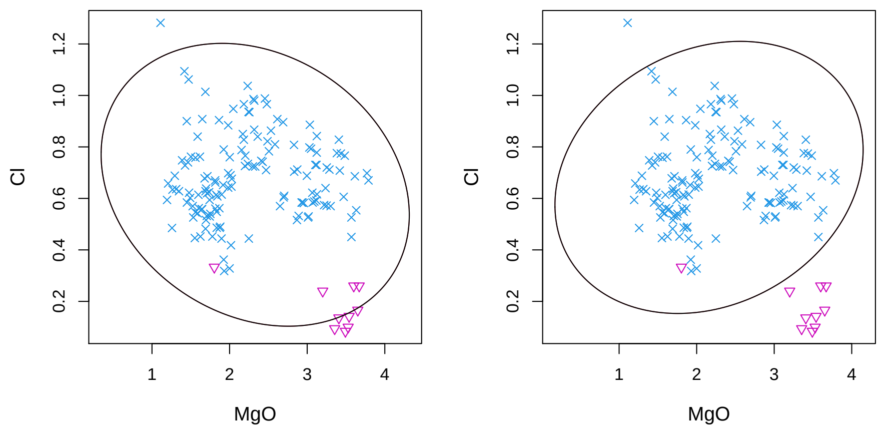


Figure 8.4: Two-dimensional glass data for two glass types, and 97.5% tolerance ellipse based on classical (left) and robust (right) estimates of location and covariance.

```
R:   library(robustbase)
R:   plot(covMcd(X))
```

An alternative visualization is shown in Figure 8.5. Here the classical (left) and robust (right) Mahalanobis distances are presented against the indexes of the observations. The second glass type obviously forms one block of observations in the data set. The outlier cutoff 2.72 is shown as a horizontal line. Of course, these plots reveal the same observations as outliers as the plots in Figure 8.4.

An advantage of the latter plots shown in Figure 8.5 is that they can also be presented if the underlying data set has more than two dimensions (because Mahalanobis distances are always univariate). Since the original data set has 13 variables, we now compute classical

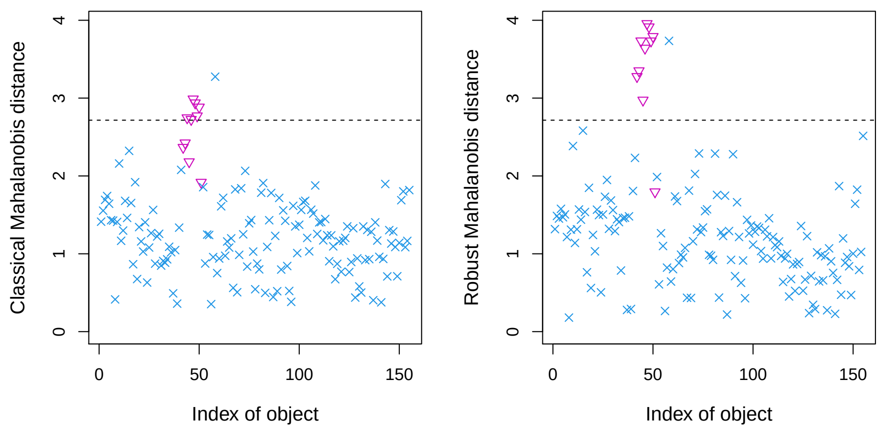


Figure 8.5: Mahalanobis distances based on classical (left) and robust (right) location and covariance estimators.

and robust Mahalanobis distances in this 13-dimensional space, whch leads to the plots in Figure 8.6. Note that the outlier cutoff value is now $\sqrt{\chi^2_{13;0.975}} = 4.97$. While the left plot shows a very similar picture as in the two-dimensional case, the right plot with the robust distances reveals data subgroups in the majority class: it seems that the first block of observations has a different data structure than the remaining observations of this glass type.

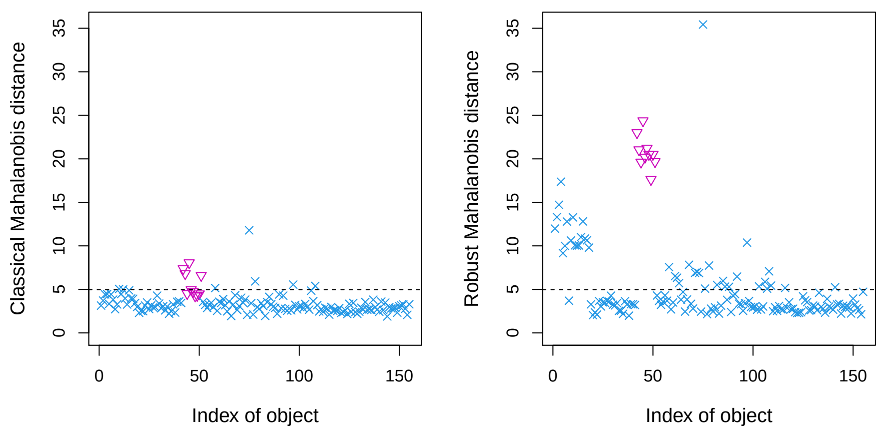


Figure 8.6: Mahalanobis distances for the 13-dimensional glass data, based on classical (left) and robust (right) location and covariance estimators.

Chapter 9

Projections of multivariate data

This chapter will present a methodology to reduce dimensionality of multivariate data. So far we have seen that data with p dimensions are not that easy to visualize, especially if p gets bigger. What we would like to "see" are groups, structures, relationships, patterns, etc. The goal here is to down-project the data to low dimension, typically to two, in order to be able to "look" at the data. Of course, this projection should contain the most relevant information.

Dimension reduction will be achieved here by using linear combinations of the original variables. Therefore, we first need to explain the concept of linear combinations.

9.1 Linear combinations of variables

Let us assume that we can get measurements of p variables $x_1, \ldots, x_p$. Once we have collected n observations, all this information will be contained in the $n \times p$ data matrix $\boldsymbol{X}$. The measurements of variable x_j are contained in $\boldsymbol{x}_j$, the j-th column of $\boldsymbol{X}$, where $j = 1, \ldots, p$.

Now consider a particular linear combination of the variables of the form

$$u = x_1 b_1 + x_2 b_2 + \ldots + x_p b_p,$$

with coefficients $b_1, b_2, \ldots, b_p$. In terms of our data matrix, this yields a vector with n observations, thus

$$\boldsymbol{u} = \boldsymbol{x}_1 b_1 + \boldsymbol{x}_2 b_2 + \ldots + \boldsymbol{x}_p b_p = \boldsymbol{X}\boldsymbol{b},$$

where $\boldsymbol{b} = (b_1, b_2, \ldots, b_p)^T$. It is common to call the coefficients *loadings*, and the values $\boldsymbol{u}$ *scores*. Since the variance of the scores would just depend on the overall size of the loadings, these coefficients are usually normed to length 1, thus $\boldsymbol{b}^T\boldsymbol{b} = \sum_{j=1}^{p} b_j^2 = 1$.

The coefficients need to be chosen according to the goal of the analysis. If we choose $b_1 = 1$ and $b_2 = \ldots = b_p = 0$, then u would be identical to x_1 (which might not be interesting). For $b_1 = \ldots = b_p = 1/\sqrt{p}$, every variable has the same contribution, and u represents something like an average (up to scaling). Thus, the coefficients could be understood as a "weight" for the variables.

Linear combinations can also be interpreted geometrically. We can view the variables $x_1, \ldots, x_p$ as the axes of a p-dimensional space, and a linear combination u would point at a particular direction in this space, defined by the contributions of the loadings. The values $\boldsymbol{u}$ can be seen as the projected data points orthogonal to this direction, see also Figure 9.1.

Generally, we could be interested in not only one projection direction, but in k, where k could be equal to p, smaller than p, or also larger than p, depending on the purpose of the analysis. Therefore, we have to adjust our notation. Let us consider the l-th linear combination, where $l = 1, \ldots, k$:

$$u_l = x_1 b_{l1} + x_2 b_{l2} + \ldots + x_p b_{lp},$$

or, in terms of data,

$$\boldsymbol{u}_l = \boldsymbol{x}_1 b_{l1} + \boldsymbol{x}_2 b_{l2} + \ldots + \boldsymbol{x}_p b_{lp} = \boldsymbol{X}\boldsymbol{b}_l,$$

with the coefficient vector $\boldsymbol{b}_l = (b_{l1}, b_{l2}, \ldots, b_{lp})^T$. All these k coefficient vectors can be collected as columns of the $p \times k$ matrix $\boldsymbol{B}$, and the score vectors can be collected in the $n \times k$ matrix $\boldsymbol{U}$. With this notation, the k linear combinations are $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{B}$.

One condition on the coefficient vectors has already been discussed: normalization to norm 1, i.e. $\boldsymbol{b}_j^T \boldsymbol{b}_j = 1$ for $j = 1, \ldots, k$. Another condition which is sometimes used is *orthogonality* of the directions, which means $\boldsymbol{b}_j^T \boldsymbol{b}_l = 0$, for $j \neq l$ and $j, l \in \{1, \ldots, k\}$. Orthogonal directions will result in scores that contain new information which has not yet been explored in other score vectors.

So far it is still unclear how the coefficients $\boldsymbol{B}$ should be selected, and even if we ask for normalized and orthogonal directions, there are still infinitely many possibilities. The choice depends on the purpose of the analysis. One purpose could be to obtain score vectors with maximum variance. This leads to "PCA", the "mother" of the multivariate methods, described in the following section.

9.2 Principal components

Principal Component Analysis (PCA) is considered as one of the most important methods in multivariate statistics. Principal components (PCs) are linear combinations of the (possibly scaled) variables, and their purpose is to represent the essential information in a lower-dimensional space. Thus, the main goal of PCA is dimension reduction.

9.2.1 Definition of principal components

Assume that we have given a centered and possibly scaled data matrix $\boldsymbol{X}$ of dimension $n \times p$. Centering means that we subtract from each column the corresponding column mean. Scaling means that we divide each column by the scale estimate of that column. The issue with centering and scaling will be discussed in more detail in Section 9.2.4.

The PCs are defined via linear combinations (what a surprise). This time, however, we will use very specific *loadings* $\boldsymbol{B}$, the *PCA loadings*, which result in very specific scores $\boldsymbol{U}$, the *PCA scores*. Once the PCA loadings are given, we obtain with the linear combination

$$\boldsymbol{U} = \boldsymbol{X}\boldsymbol{B}$$

immediately the PCA scores.

How shall we select these loadings? What do we want to achieve for the scores? What we want is a scores matrix $\boldsymbol{U}$ of the same dimension $n \times p$ as $\boldsymbol{X}$, and thus just a different representation of our (centered and possibly scaled) data. Since the goal is dimension reduction, we would like that the most relevant information is already contained in the first few columns of $\boldsymbol{U}$. "Relevant" refers to "explained variance", and this is the key to defining the PCA loadings and scores.

Denote the columns of $\boldsymbol{U}$ as $\boldsymbol{u}_1, \dots, \boldsymbol{u}_p$ (they are often simply called principal components), and the columns of $\boldsymbol{B}$ as $\boldsymbol{b}_1, \dots, \boldsymbol{b}_p$. These vectors will be uniquely determined by the following optimization problem (under constraints):

- The first loadings vector $\boldsymbol{b}_1$ is chosen such that the variance of $\boldsymbol{u}_1 = \boldsymbol{X}\boldsymbol{b}_1$ is maximized. Since this variance should not depend on the scale of $\boldsymbol{b}_1$, we constrain this vector to be normalized to length 1, thus $\boldsymbol{b}_1^T\boldsymbol{b}_1 = 1$.
- The second loadings vector $\boldsymbol{b}_2$ is chosen such that the variance of $\boldsymbol{u}_2 = \boldsymbol{X}\boldsymbol{b}_2$ is maximized, under the contraints $\boldsymbol{b}_2^T\boldsymbol{b}_2 = 1$ and $\boldsymbol{b}_2^T\boldsymbol{b}_1 = 0$. The latter constraint asks for orthogonality to the previous direction $\boldsymbol{b}_1$.
- The j-th loadings vector $\boldsymbol{b}_j$ (for $2 < j \leq p$) is chosen such that the variance of $\boldsymbol{u}_j = \boldsymbol{X}\boldsymbol{b}_j$ is maximized, under the contraints $\boldsymbol{b}_j^T\boldsymbol{b}_j = 1$ and $\boldsymbol{b}_j^T\boldsymbol{b}_l = 0$, for $1 \leq l < j$. The latter constraint asks for orthogonality to all previous directions.

Figure 9.1 shows for a very simple data set of 10 observations in two dimensions the first PC: The direction is determined by the loadings vector $\boldsymbol{b}_1$, and the scores $\boldsymbol{u}_1$ are the data points orthogonally projected on this direction. These new univariate data points have the largest possible variance among all possible projections. The second PC would be orthogonal to the first one, and cover the remaining variance.

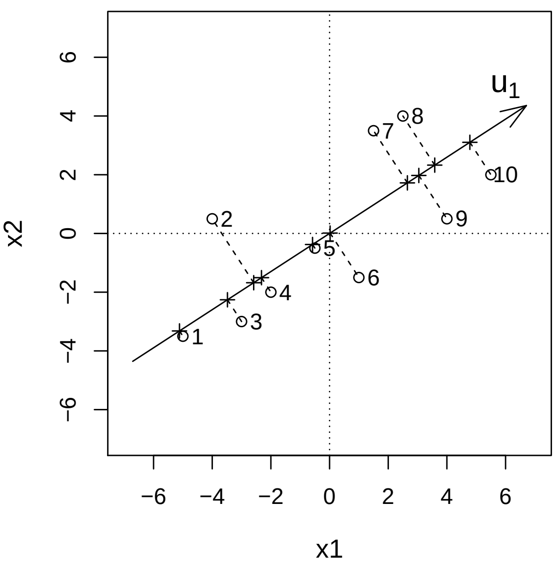


Figure 9.1: First principal component of a two-dimensional artificial data set.

As mentioned above, the PCs build a new orthogonal coordinate system in which the data set is represented. If all p PCs are used, we essentially have a rotation of the original coordinates. However, the goal of dimension reduction is to express the data information in fewer coordinates. Since the PCs are ordered according to variance, we could use the first $k < p$ PCs to express the most essential information in a lower-dimensional space. A discussion about the choice of k can be found in Section 9.2.3.

9.2.2 Algorithm to compute PCs

The above constrained maximization problem already outlines how we could compute the PCs. However, what is now the solution for the columns $\boldsymbol{b}_j$ to determine the PCA scores $\boldsymbol{u}_j$? Do we have to scan the whole p-dimensional space in order to identify these direction vectors? This would be quite time consuming. Below we will see that there is a much easier way to obtain the solutions.

Maximizing variance under constraints can be easily formulated by a Langrange problem. If we talk about *variance*, we should also mention which estimator of variance we have in mind. We could take the sample variance, or a robust estimator. Obviously, the sample variance could be sensitive to outliers, which is not the case for a robust variance estimator like the (squared) MAD. For simplicity and generality we will in the following simply denote this estimator by "Var".

The j-th PC is defined as $\boldsymbol{u}_j = \boldsymbol{X}\boldsymbol{b}_j$, for $1 \leq j \leq p$, and we want to maximize

$$\mathrm{Var}(\boldsymbol{u}_j) = \mathrm{Var}(\boldsymbol{X}\boldsymbol{b}_j) = \boldsymbol{b}_j^T \mathrm{Cov}(\boldsymbol{X})\boldsymbol{b}_j.$$

The latter equality can easily be checked by employing the properties of variance and covariance, and now our data matrix is involved in terms of $\mathrm{Cov}(\boldsymbol{X})$, which refers to the covariance matrix, where we denote with "Cov" a specific covariance estimator. Note that in Section 8.1.1 we have discussed different options, such as the sample covariance matrix $\boldsymbol{S}$, or the robust MCD covariance. We still have the constraint $\boldsymbol{b}_j^T\boldsymbol{b}_j = 1$, which needs to be considered in the Langrange problem,

$$\phi_j = \boldsymbol{b}_j^T \mathrm{Cov}(\boldsymbol{X})\boldsymbol{b}_j - \lambda_j(\boldsymbol{b}_j^T\boldsymbol{b}_j - 1) \qquad \text{for } j = 1, \ldots, p$$

with the Langrange multipliers λ_j. One can show that including also the orthogonality constraints in the Lagrange problem would not change the solutions.

In order to identify the solutions for $\boldsymbol{b}_j$, we need to compute the partial derivatives

$$\frac{\partial \phi_j}{\partial \boldsymbol{b}_j} = 2\mathrm{Cov}(\boldsymbol{X})\boldsymbol{b}_j - 2\lambda_j\boldsymbol{b}_j = \boldsymbol{0}.$$

and set them equal to zero. This is the same as

$$\mathrm{Cov}(\boldsymbol{X})\boldsymbol{b}_j = \lambda_j\boldsymbol{b}_j \qquad \text{for } j = 1, \ldots, p,$$

which is just the form of an eigenvalue-eigenvector problem: the $\boldsymbol{b}_j$ are the eigenvectors of $\mathrm{Cov}(\boldsymbol{X})$ to the eigenvalues λ_j. This is now the final answer to the question how we shall select the matrix $\boldsymbol{B}$.

Suppose that we would have a data matrix $\boldsymbol{X}$ in `R`. The code to compute loadings and scores would then be:

```
X.cs <- scale(X)    # here centering and scaling is done
S <- cov(X)         # computes the sample covariance matrix
evv <- eigen(S)     # is doing the eigen-decomposition
B <- evv$vectors    # gives the PCA loadings
U <- X.sc%*%B       # gives the PCA scores
```

The Langrange multipliers λ_j turn out to be the eigenvalues, and they have an interesting role:

$$\mathrm{Var}(\boldsymbol{u}_j) = \boldsymbol{b}_j^T \mathrm{Cov}(\boldsymbol{X})\boldsymbol{b}_j = \boldsymbol{b}_j^T\lambda_j\boldsymbol{b}_j = \lambda_j\boldsymbol{b}_j^T\boldsymbol{b}_j = \lambda_j$$

Thus, the eigenvalues are equal to the maximized variances of the PCs. If we look at

```
    evv$values   # eigenvalues
```

we obtain the eigenvalues, and we can see that they are sorted in decreasing order. This means that indeed the first column of $\boldsymbol{U}$ has the largest variance, the second has the second largest, and so on. We can verify the result also with

```
    apply(U,2,var)  # variances of the PCs, equal to the eigenvalues
```

which computes the sample variances of the PCA scores. We could also easily verify, that the loadings have length 1, and that different loading vectors are orthogonal to each other.

In practice one would use a more convenient function to compute the PCA solution:

```
    X.pca <- princomp(X,cor=TRUE) # PCA computed for centered and scaled data
```

9.2.3 Number of relevant PCs

The goal of PCA is to reduce dimensionality, and at the same time to keep the loss of information small. Since the variances of the PCs are sorted in decreasing order, we can easily compute the variance expressed by the first k PCs as $\lambda_1 + \ldots + \lambda_k$. The total variance is $\lambda_1 + \ldots + \lambda_p$. Therefore, the proportion of explained variance by the first k PCs is:

$$\frac{\lambda_1 + \ldots + \lambda_k}{\lambda_1 + \ldots + \lambda_p}$$

The PC number versus this proportion is visualized by the so-called `scree plot`, see Figure 9.2 (left) for an example. The optimal number k of PCs could then be selected at that point before the curve "flattens out", which is indicated here by the straight line. Thus, the first two PCs could be our choice for dimension reduction.

How much variance is explained by these two PCs? This is visualized in the right plot of Figure 9.2, with the PC number versus the cumulative explained variance. Often the number of relevant PCs is selected so that around 80% of the variance is explained, which would be the case in our example. However, the desired percentage really depends on the subsequent purpose of the analysis. For exploratory data analysis purposes we might also be satisfied with a (much) lower percentage, for modeling we might need a (much) higher one.

9.2.4 Centering and scaling

As discussed previously, centering and scaling the data columns might be important. On the other hand, for the described procedure, centering will not change the results, because the solutions are derived by an eigen-decomposition of the covariance matrix, and the covariance matrix would not change after centering. Centering will only be relevant when we would use a different procedure to compute PCs, such as SVD (singular value decomposition).

Scaling, however, always makes a difference. After scaling, the variances of the data columns are equal to one, and thus the scaled variables are statistically comparable. If the variances would be very different – which is true for many data sets – they would impact the linear combinations differently. Remember that we maximize $\mathrm{Var}(\boldsymbol{u}_j) = \mathrm{Var}(\boldsymbol{x}_1 b_{j1} + \ldots + \boldsymbol{x}_p b_{jp})$, and those variables $\boldsymbol{x}_l$ with the biggest variance would receive the highest coefficients since they drive the maximum most. In that sense, bringing the variables to the same variance is "fair" to the variables.

Note that the covariance matrix of scaled variables is the same as the correlation matrix (where covariances are divided by the square-roots of the variances). Thus, what we effec-

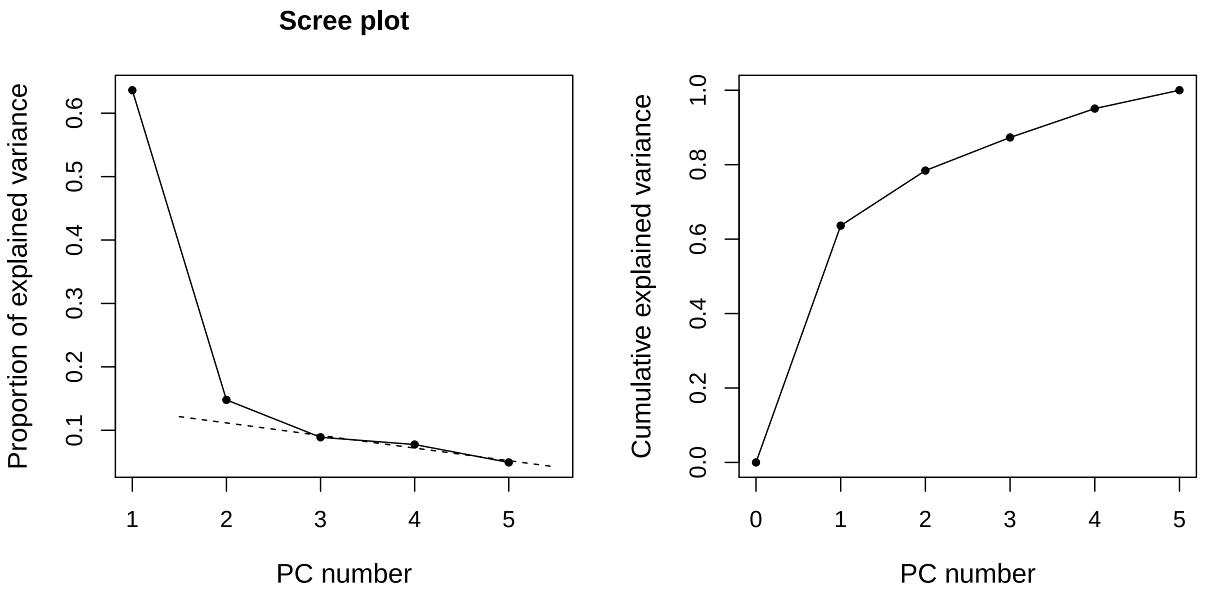


Figure 9.2: Scree plot (left) and cumulative proportion of explained variance (right). The data set is obtained from `data(scor,package="bootstrap")`.

tively do is to eigen-decompose the correlation matrix rather than the covariance matrix. This explains the argument `cor=TRUE` inside the `princomp()` function.

9.2.5 Normal distribution and outliers

Personally, I can't remember any distributional assumption when formulating the PCA problem. Did we assume multivariate normal distribution? No, the PCA problem boiled down to a mathematical optimization problem under constraints – nothing to do with statistics (one could say).

On the other hand, we know that the solution of the PCA problem boils down to an eigen-decomposition of the covariance matrix of $\boldsymbol{X}$. If we use the sample covariance matrix $\boldsymbol{S}$ as an estimator, then we know that this is sensitive with respect to outliers, and that implicitly we need to assume multivariate normality.

Figure 9.3 (left) shows a bivariate data set with body and brain weight of different animals (sorry, also the human is included). The data set is available as `data(Animals2)` from the `R` package `robustbase`. The plot already shows the centered and scaled data, as well as the direction of the first PC. With this principal component, 52.6% of the total variance are explained. Both body and brain weight are very skewed as there are very differnt animals included (mouse, horse, brachiosaurus, etc.). Especially, the heavy animals appear as outliers, and they influence quite a lot the sample covariance estimation. The right plot shows the scaled and log-transformed variables, again with the first PC, which now explains 93.8%. The data seem to follow essentially a bivariate normal distribution (with some outliers). We can conclude that a transformation of the data to approach normality had clear advantages in terms of dimension reduction, here from two to one dimension.

Since we seem to also have an outlier issue with the data set used in Figure 9.3, we could think about more robust covariance estimators, such as the MCD covariance estimator which is less sensitive to data outliers, see Section 8.1.1. Using this estimated covariance matrix for the eigen-decomposition of the PCA problem will yield loadings and scores that are not

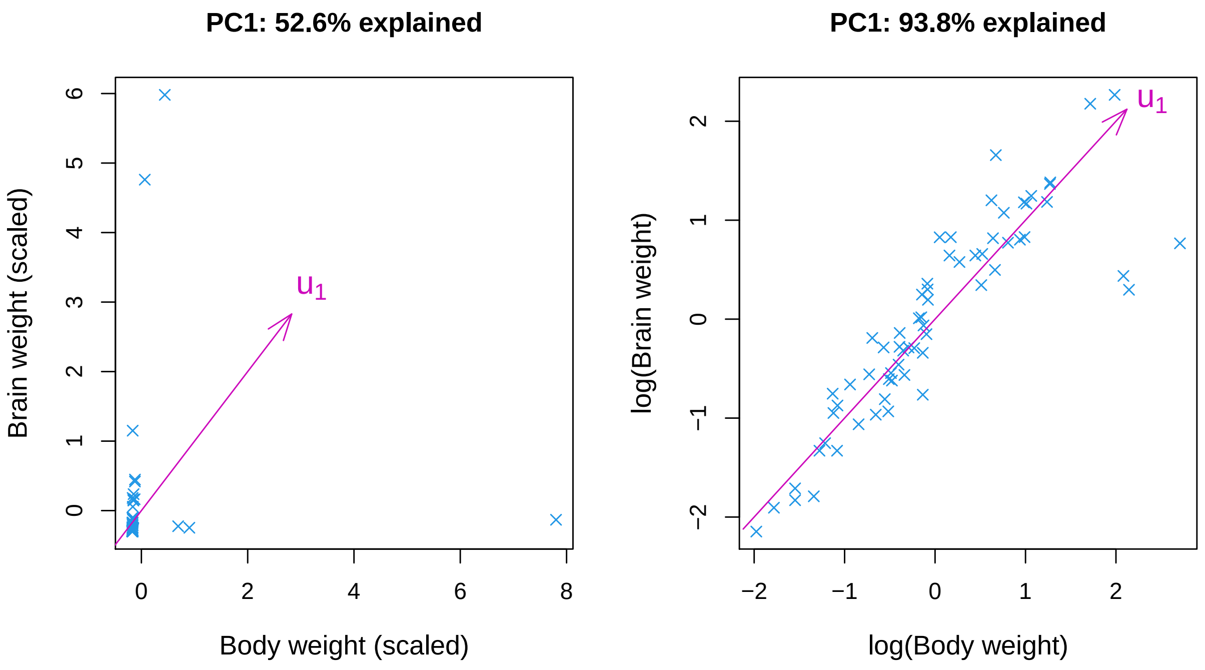


Figure 9.3: Body and brain weight of various animals: left only scaled, right log-transformed and scaled. The first PC and the resulting explained variance are indicated in the plots.

affected by the outliers. This is done in Figure 9.4. The left plot shows the first PC with the robustly scaled data (89.2% explained variance), and the right plot contains the first PC for the robustly scaled log-transformed data (97.8% explained variance). In both plots, the red symbols indicate outliers identified with the robust Mahalanobis distances. For the latter PCA, the `R` code is shown below.

```
library(robustbase)
data(Animals2)
x.mcd <- covMcd(log(Animals2),cor=TRUE)
xlog <- scale(log(Animals2),center=x.mcd$center,scale=sqrt(diag(x.mcd$cov)))
xlog.pc <- princomp(xlog,cor=TRUE,covmat=x.mcd)
```

9.2.6 Visualizing the results, biplots

The most interesting results of PCA are contained in the first k columns of the loadings matrix $\boldsymbol{B}$ and scores matrix $\boldsymbol{U}$. We could simply plot this information.

Example: For the scree plot in Figure 9.2 we have used the data set `scor` from the package `bootstrap` with information from 88 students about their results in the subjects ME (mechanics), AG (analytical geometry), LA (linear algebra), AN (analysis), and ES (elementary statistics). In each subject they could attain up to 100 points. From the scree plot and the plot of the cumulative variances we know that $k = 2$ components might be sufficient for data exploration. We could then simply plot the resulting two PCs with the scores $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, and the loadings $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$.

```
data(scor,package="bootstrap")
pca <- princomp(scor,cor=TRUE)  # PCA for scaled data
```

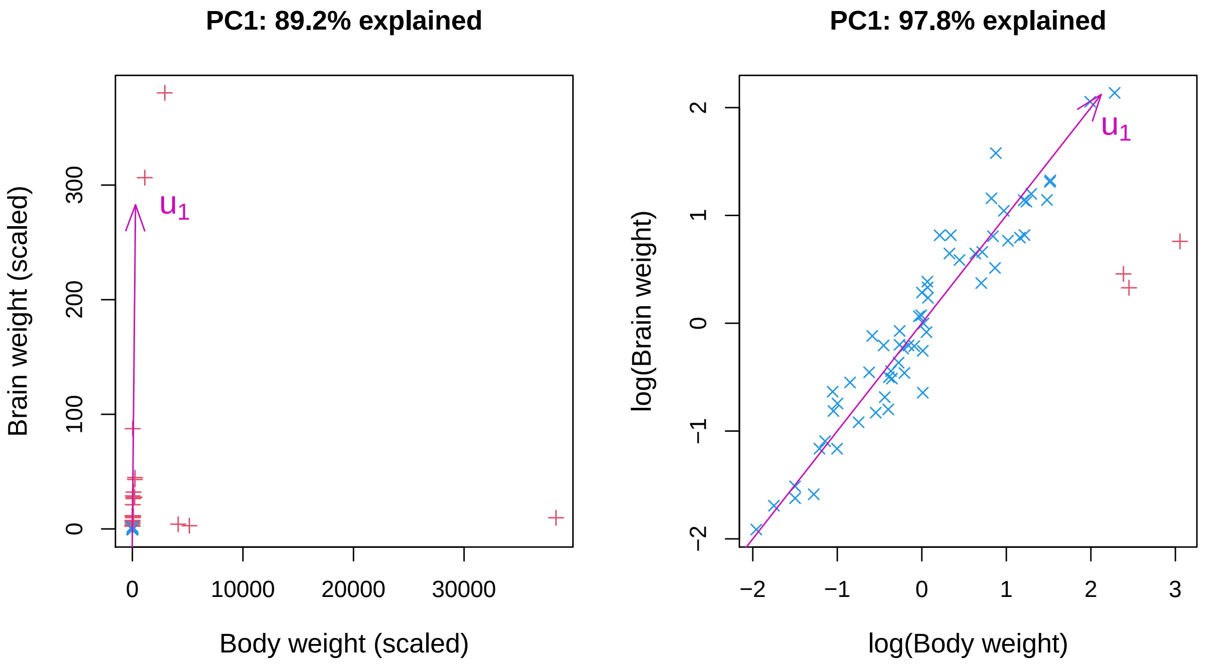


Figure 9.4: Body and brain weight of various animals: left only scaled, right log-transformed and scaled. The first robust PC and the resulting explained variance are indicated in the plots. Outliers are shown by red symbols.

```
plot(pca$scores[,1:2])          # plot scores
plot(pca$loadings[,1:2])        # plot loadings
```

According to the loadings plot in Figure 9.5 (right) we see that all variables contribute positively with about the same magnitude to PC1, and thus PC1 can be interpreted as an average result in the subjects. PC2 has positive contributions from subjects with more emphasis on geometry, and negative contributions with subjects more related to mathematical skills. The scores plot (left) reveals that the student IDs from 1 to 88 seem to be ordered along PC1, thus according to their average result, with small numbers referring to the top students. PC2 is not so easy to interpret, but we can see, for example, that students 66 and 76 with intermediate average results must bee quite good in ME and AG, but poor in AN and ES. Indeed, the original data show:

```
   ME AG LA AN ES
66 59 53 37 22 19
76 49 50 38 23  9
```

It is possible to combine both plots from Figure 9.5 in one single plot, which is called **biplot**. The term “bi” refers to representing both loadings and scores in one plot. For this plot, scores and loadings are rescaled, which leads to interesting properties:

- The orthogonal projections of the observations onto the variables (represented by the arrows) approximate the (centered and scaled) data values.
- The cosines between the angles of the arrows for the variables approximate the correlations between the variables.

We talk about “approximation” because we lost some information; in this example about 20%, because we reduced dimensionality from 5 to 2.

```
biplot(pca)                     # biplot with the princomp object
```

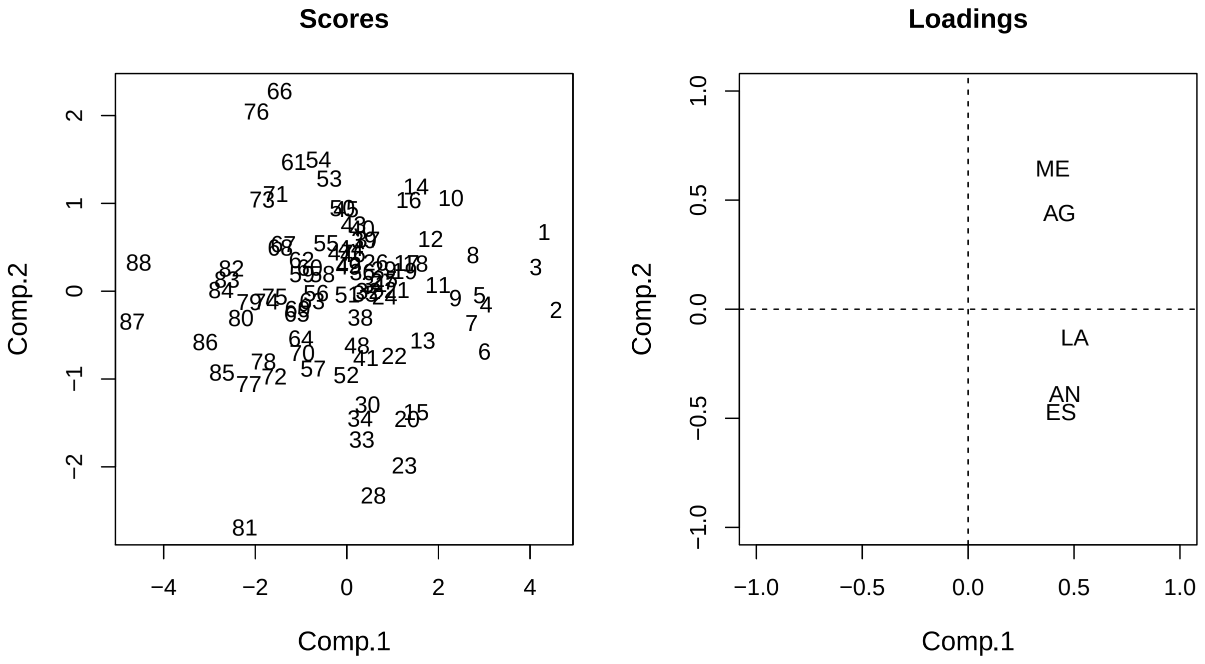


Figure 9.5: *Scores* (left) and *loadings* (right) of the first two PCs for the data set `data(scor,package="bootstrap")`.

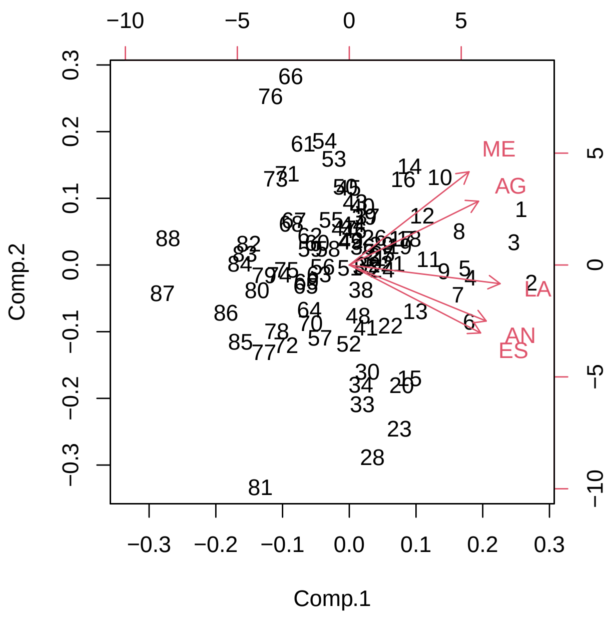


Figure 9.6: Biplot of the first two PCs for the data set `data(scor,package="bootstrap")`.

Chapter 10

Some multivariate statistical methods

So far we mentioned regression and principal component analysis in the context of analyzing multivariate data. However, there are more potential problem settings, and of course we cannot cover all of them. Thus, we will focus in this chapter on two very important multivariate methods: on cluster analysis, and on discriminant analysis.

10.1 Cluster analysis

The term “cluster” has the meaning of a “concentrated” group. Usually, when we talk about cluster analysis, we want to identify observations which are similar to each other, and thus “concentrated” in the multivariate space. The task is thus to automatically group the observations into “homogeneous” groups. The observations within a group should be similar to each other, while observations from different groups are supposed to be dissimilar (“heterogeneous”). Sometimes, cluster analysis is applied to the variables, and then we would search for groups of similar variables. This has been done in Chapter 7 for some techniques of multivariate graphical displays (boxes, trees).

The similarity of observations can be determined by a distance measure. Basically, we could consider here any distance measure discussed in Section 8.2. Most of the distance measures depend on the scale of the variables, and thus *centering and scaling* the variables is often an important first step in cluster analysis.

There are several challenges in cluster analysis:

- We do not know if a grouping structure in indeed inherent in the data set.
- We do not know how many groups there could be.
- We do not know which observations could belong to which groups.

In order to make some progress, we could simply assume that there is a cluster structure, we could assume that the data set contains k clusters, and then we could try to estimate the assignment of the observations to the clusters. In the end, we should also come up with some measure which provides an idea about the “quality” of the identified clusters; this is called a *cluster validity measure.*

There are different procedures to construct clusters. Some important ones are:

- **Partitioning methods:** The observations are grouped into k clusters, and thus every observation is assigned to exactly one cluster (disjoint grouping of the observations).

- **Hierarchical clustering methods:** A hierarchy of partitions is constructed, where the number of clusters is varied from 1 to n, where n is the number of observations, or from n to 1. The former procedure is called *divisive*, and the latter *agglomerative* (which is much more frequently used).
- **Fuzzy clustering:** Every observation is assigned "proportionally" to each of the k clusters. The degree of assignment is determined by cluster membership coefficients which are in the interval $[0, 1]$ and sum up to 1 per observation.
- **Model-based clustering:** This also results in a partitioning of the observations into k clusters, but the shape of the clusters is modeled by a multivariate normal distribution with a certain mean and covariance.

In the following we assume to have an $n \times p$ data matrix $\boldsymbol{X}$, and the task is to cluster the observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$.

10.1.1 Partitioning methods

Suppose we want to partition the observations into k distinct clusters. Further, denote I_j as the index set containing the indexes of the observations of the j-th cluster, and n_j denotes the number of indexes in I_j ($j = 1, \ldots, k$). The index set is thus of the form $I_j = \{i_1, i_2, \ldots, i_{n_j}\}$, and thus the j-th cluster contains the observations $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_{n_j}}$. For partitions we have that $n_1 + n_2 + \ldots + n_k = n$.

The most popular algorithm for the construction of partitions is the **k-means** algorithm. The input parameter is the desired number k of clusters. The k-means algorithm uses so-called centroids or cluster centers, which simply can be taken as the arithmetic means of the observations per cluster:

$$\bar{\boldsymbol{x}}_j = \frac{1}{n_j} \sum_{i \in I_j} \boldsymbol{x}_i \quad \text{for} \quad j = 1, \ldots, k \tag{10.1}$$

The objective function for k-means is

$$\sum_{j=1}^{k} n_j \sum_{i \in I_j} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j\|^2 \longrightarrow \min, \tag{10.2}$$

and it has to be minimized in order to find the resulting index sets with the cluster assignments and the corresponding cluster centers. The algorithm follows an iterative scheme, where in every step of the iterations the index sets I_j will (slightly) change. After convergence we obtain the final index sets $I_1^*, \ldots, I_k^*$, and therefore the final assignments of the n observations to the k clusters. The iterative algorithm is commonly initialized by a random selection of k observations, which form the k cluster centers at the beginning. Depending on this random selection, one could end up with diffrent cluster solutions after convergence. This means that we might end up in local optima, and for approximating the global optimum, one can start the algorithm several times and take that solution with the smallest value of the objective function.

```
# R code k-means
Xs <- scale(X)         # X is the data matrix, which is first centered and scaled
res <- kmeans(Xs,3)    # k=3 is the desired number of clusters
str(res)               # shows the content of the output object
res$cluster            # assignments of the observations to the clusters
```

An algorithm which is very similar to k-means is called **PAM** (Partitioning Around Medoids); here the cluster centroids are robustly estimated by medians.

```
# R code PAM
Xs <- scale(X)        # X is the data matrix, which is first centered and scaled
library(cluster)
res <- pam(Xs,3)      # k=3 is the desired number of clusters
str(res)              # shows the content of the output object
res$clustering        # assignments of the observations to the clusters
plot(res)             # plots for diagnostics
```

The decision on an appropriate number k of clusters is usually done with a validity measure, which will be discussed later. The procedure is then to run k-means or PAM for different values of k, and then select that k which gives the best value of the validity measure. A validity measure can also be useful if there are different solutions for the same k.

10.1.2 Hierarchical clustering

Here we will describe the procedure for agglomerative clustering in hierarchies. At the beginning, every observation forms an own cluster; these n clusters are called *singletons*. In the next step, those sigletons with smallest distance to each other are merged, and we obtain $n-1$ clusters. As soon as we want to merge non-sigleton clusters, we need to have a distance measure between clusters (and not just between single observations), which is given as follows.

Let two clusters be given by the index sets I_j and $I_{j'}$. The following definitions of distance measures between clusters are very popular, and they also give the name to the clustering method:

- **Complete linkage:** $\max_{i \in I_j, i' \in I_{j'}} d(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$
- **Single linkage:** $\min_{i \in I_j, i' \in I_{j'}} d(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$
- **Average linkage:** $\text{average}_{i \in I_j, i' \in I_{j'}} d(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$
- **Centroid method:** $d(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_{j'})$
- **Ward method:** $d(\bar{\boldsymbol{x}}_j, \bar{\boldsymbol{x}}_{j'}) \frac{\sqrt{2 n_j n_{j'}}}{\sqrt{n_j + n_{j'}}}$

As a distance $d(\cdot, \cdot)$ we can use any distance measure from Section 8.2, e.g. the Euclidean distance.

The different algorithms (distance measures) typically also lead to different cluster results. For example, *single linkage* is known for the "'chaining effect", which means that clusters typically grow by adding small clusters in every step.

Figure 10.1 illustrates for two clusters (presented by ellipses) the distances *complete* and *single linkage*. Thus, in a particular step of the hierarchical clustering algorithm, those two clusters are merged which have the smalles distance.

For the example data set in Figure 10.1 we can see in Figure 10.2 the whole cluster hierarchy, built up with *complete linkage*. In contrast to k-means, this algorithm does not result in any randomness – it is always uniquely given by the minimization problem.

The results in Figure 10.2 can be visualized more effectively in the so-called **dendrogram**. The vertical direction shows the distance (e.g. *complete linkage*), which grows step-by-step.

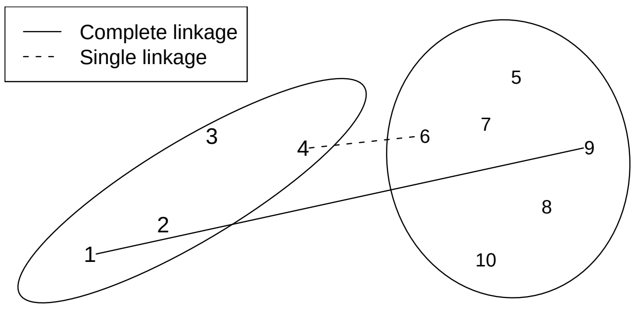


Figure 10.1: Illustration of the *complete* and *single linkage* distance (here based on the Euclidean distance) for two clusters.

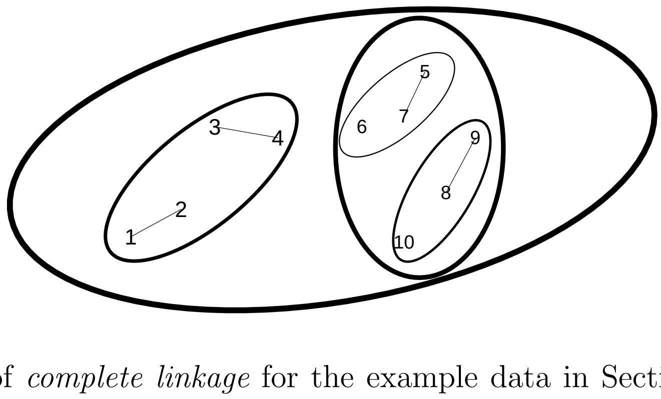


Figure 10.2: Result of *complete linkage* for the example data in Section 10.1. First, every observation forms its own cluster. These are then merged step-by-step, until all observations are joined in one single cluster.

Horizontal lines refer to connections of clusters at the corresponding distance. The observations are arranged in order to avoid intersections between the lines. Thus, we can see the hierarchy by reading the dendrogram from the bottom to the top.

Figure 10.3 shows on the left the dendrogram as a result from *complete linkage*, and on the right that from *single linkage*. While *complete linkage* very clearly reveals two clusters (these would be merged only at a very large distance), this is not so visible for *single linkage*. This visual representation can thus be used to get an idea about the number k of clusters which is inherent in the data. A horizontal cut at the corresponding distance allows to assign the observations to the k clusters.

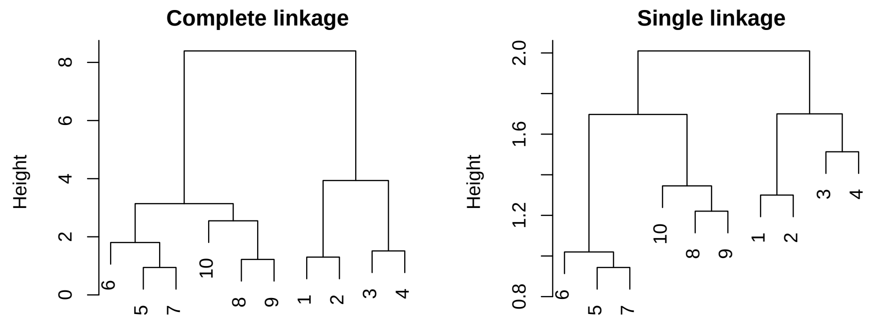


Figure 10.3: Dendrogram for the data set from Figure 10.1. Left the solution for *complete linkage*, see also Figure 10.2, right the solution for *single linkage*.

```
# R code hierarchical clustering
Xs <- scale(X)            # X is the data matrix, which is first centered and scaled
res <- hclust(dist(Xs))  # default is complete linkage and Euclidean distance
plot(res)                 # show the dendrogram
cl <- cutree(res,3)       # assigns the observations to 3 clusters
```

10.1.3 Fuzzy clustering

In contrast to a "hard" assignment of the observatiions to the clusters, as done in the previous sections, fuzzy clustering allows for a "soft" assignment – it is often not so clear if an observation belongs to one cluster or falls "in between" several clusters. More specifically, partitions assign each of the n observations to precisely one of the k clusters. For fuzzy clustering we proportionally distribute the observations over the k clusters. This is done by a membership coefficient u_{ij}, for $i = 1, \dots, n$ and $j = 1, \dots, k$, which is in the interval $[0, 1]$. Since we want a proportional assignment, we ask for $\sum_{j=1}^{k} u_{ij} = 1$, for all i.

The clustering algorithm thus needs to estimate the matrix with coefficients u_{ij}. The number of clusters k is – similar to k-means – given by the user. Note that the result of fuzzy clustering can always be converted to a hard clustering result, by simply assigning an observation to that cluster for which its membership coefficient is the biggest.

The most widely used algorithm is the *fuzzy c-means* algorithm. The objective function is similar to k-means:

$$\sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij}^2 \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_j\|^2 \longrightarrow \min, \tag{10.3}$$

with the cluster centroids

$$\tilde{\boldsymbol{x}}_j = \frac{\sum_{i=1}^n u_{ij}^2 \boldsymbol{x}_i}{\sum_{i=1}^n u_{ij}^2} \quad \text{for} \quad j = 1, \ldots, k. \tag{10.4}$$

Again, this problem is solved by an iterative algorithm, where in every step the coefficients u_{ij} are updated. Similar to k-means, the algorithm is initialized randomly, which can lead to different results for different runs.

```
# R code fuzzy clustering
Xs <- scale(X)           # X is the data matrix, which is first centered and scaled
library(e1071)           # Why on earth this strange name?
res <- cmeans(Xs,3)      # fuzzy c-means with 3 clusters
str(res)                 # shows the content of the output object
res$cluster              # hard cluster assignments
res$membership           # matrix with the membership coefficients
```

10.1.4 Model based clustering

In contrast to the previous methods, here we also try to model the shape of the clusters. (Note that k-means would typically identify spherically shaped clusters.) We will assume that the observations of the j-th cluster are realizations of a p-dimensional normal distribution with density function given by

$$\boldsymbol{\phi}_j(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^p \det(\boldsymbol{\Sigma}_j)}} \exp\left\{ -\frac{(\boldsymbol{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)}{2} \right\}, \tag{10.5}$$

with mean $\boldsymbol{\mu}_j$ and covariance $\boldsymbol{\Sigma}_j$. In fact, the underlying data distribution is considered as a mixture of k such normal distributions, with prior probabilities p_j, summing up to 1.

The clustering algorithm needs to estimate the parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ of the normal distributions, as well as the prior probabilities, reflecting the class proportions. Once these parameters are estimated, resulting in $\hat{p}_j$ and $\hat{\phi}_j(\boldsymbol{x}_i)$ one can compute an "expected" class assignment by $\hat{p}_j\hat{\phi}_j(\boldsymbol{x}_i)$ for every observation to every class, and assign the i-th observation to that class, for which this value is the biggest.

Once the cluster assignments are estimated, one can re-estimate the parameters of the normal distributions as well as the prior probabilities. This procedure, iterating the two steps, is called EM (expectation maximization) algorithm.

Particularly the estimation of the $p \times p$ matrices $\boldsymbol{\Sigma}_j$, for $j = 1, \ldots, k$, can lead to difficulties and instabilities, since this requires a lot of data information (especially if p is bigger). A way out is to simplify the problem by assuming, for example, that the covariances of all clusters are identical, and thus the cluster shapes are the same. One can think about various different options of simplifications, which are discussed in the following. The main idea is to reduce the number of parameters of the covariance matrices to be estimated.

The very simplest option would be $\boldsymbol{\Sigma}_j = \sigma^2 \boldsymbol{I}$, for $j = 1, \ldots, k$. Here, $\boldsymbol{I}$ is the identity matrix, and σ^2 is a parameter for the variance. Thus, all clusters would be spherical, with the same radius in all dimensions. We would only have to estimate one parameter σ.

A less restrictive assumption would be $\boldsymbol{\Sigma}_j = \sigma_j^2 \boldsymbol{I}$, for $j = 1, \ldots, k$. Still, the clusters would be spherical, but with different sizes acoording to the variances σ_j^2.

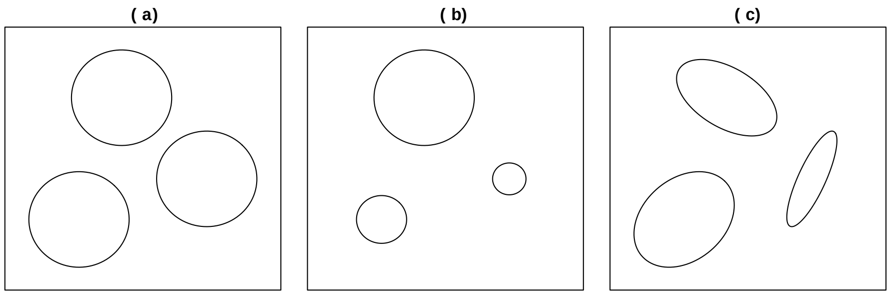


Figure 10.4: Different restrictions on the covariances of the three clusters: (a) $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}_3 = \sigma^2\boldsymbol{I}$; (b) $\mathbf{\Sigma}_j = \sigma_j^2\boldsymbol{I}$, for $j = 1, 2, 3$; (c) all $\mathbf{\Sigma}_j$ can be different from each other.

Figure 10.4 illustrates different restricted models – see figure caption for an explanation.

One practical advantage of model-based clustering is that the data do not have to be scaled, because the variances in the models can adjust to the data scale.

In the following code example we use the function `Mclust()` from the package `mclust` to cluster the iris data set. This algorithm allows to provide a range of different numbers of clusters, here from 3 to 9:

```
# R code for model-baseed clustering
data(iris)                       # iris data, only columns 1-4 are used
library(mclust)
res <- Mclust(iris[,1:4],3:9)   # solutions for 3 to 9 clusters
plot(res)                        # diagnostic plots
str(res)                         # look at the output object
res$classification               # cluster assignments for the best solution
```

Results are shown in Figure 10.5. The left plot shows the so-called BIC (Bayesian Information Criterion) values for the cluster solutions with different numbers of clusters k (horizontal axis), as well as for different model restrictions. BIC values mimic a model prediction measure, and should thus inform about the combination of k and model restriction that gives the best possible model for our problem setting (maximum of all BIC values). The legend refers to the models with the different restrictions. For example, model "EII" is the simplest model, with covariances $\mathbf{\Sigma}_j = \sigma^2\boldsymbol{I}$. Thee models are then more and more complex, up to the most general model "VVV", with cluster-specific covariances $\mathbf{\Sigma}_j$. In this case, the best model is for $k = 3$ clusters and model "VEV".

The right plot in Figure 10.5 shows a scatterplot matrix of the data with the observations assigned to the $k = 3$ clusters, and ellipses indicated the covariance structure.

10.1.5 Cluster validity measures

A cluster validity measure should help with the decision about an appropriate number of clusters k, but also for selecting an appropriate cluster solution as the result of possibly different cluster alsgorithms. Unfortunately, there are at least as many proposals for validity measures as clustering algorithms. The BIC criterion from model-based clustering would be one option.

The aim of cluster analysis is to end up with homogeneous clusters, i.e. the observations assigned to the same cluster should be very similar to each other, but at the same time to achieve high heterogeneity between different clusters, i.e. observations assigned to different

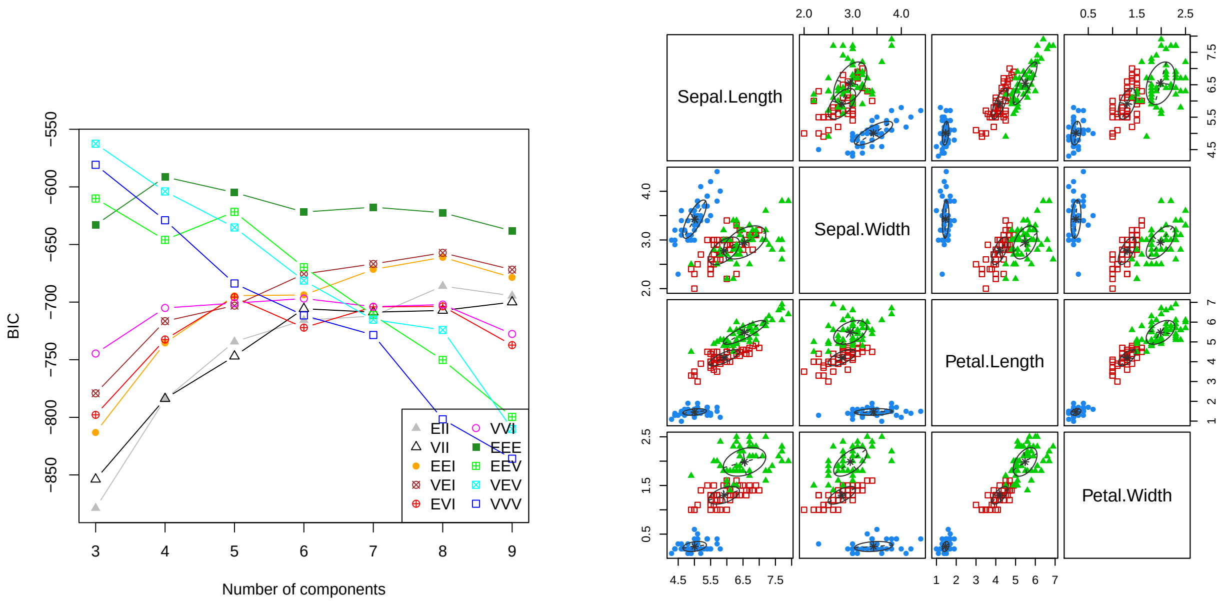


Figure 10.5: Results of model-based clustering with the function `Mclust()`: left the BIC values for the desired numbers of clusters and the different model restrictions; right the scatterplot matrix with assignments and ellipses for the best solution according to the BIC.

clusters should be dissimilar to each other. Measures of *homogeneity* could be based on the maximum, minimum, or average distance of all observations of a cluster, or on the spread (variance) of the observations of a cluster. The latter is expressed e.g. by the *within-cluster sum-of-squares*

$$W_k = \sum_{j=1}^{k} \sum_{i \in I_j} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j\|^2, \tag{10.6}$$

see also Equations (10.1) and (10.2). The value of W_k should preferably be small, but this also depends on k; the more clusters we choose, the smaller will be the variance of the observations within a cluster.

For the *heterogeneity* we can use the cluster distance measures *complete linkage*, *single linkage*, etc., or we can consider the *between-cluster sum-of-squares*

$$B_k = \sum_{j=1}^{k} \|\bar{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}\|^2, \quad \text{with} \quad \bar{\boldsymbol{x}} = \frac{1}{k}\sum_{j=1}^{k} \bar{\boldsymbol{x}}_j. \tag{10.7}$$

The values of B_k should preferably be big, but again this depends on the number k of clusters.

The **Calinski-Harabasz index** is a normalized ratio of these quantities,

$$\mathrm{CH}_k = \frac{B_k/(k-1)}{W_k/(n-k)}.$$

Another proposal, the **Hartigan-Index**, is defined as

$$\mathrm{H}_k = \log \frac{B_k}{W_k}.$$

In order to select an appropriate value of k it is recommended to plot these indexes against the values of k. For Calinski-Harabasz we would take that k which gives the biggest value.

For Hartigan we would take that k where we can see a “knee” in the plot, thus where the increase flattens out. We admit that these instructions might not be entirely clear in a specific application, because the “knee” might hardly be visible. As mentioned above, there are many alternative validity measures which could be used instead. For instance, the package `cluster` contains the *average silhouette width*, which is very useful in various applications.

10.2 Discriminant analysis

In cluster analysis we basically did not know if there is a grouping structure in the data, and we also did not know the number of clusters. The setting in discriminant analysis is very different: Here we have observations which belong to different groups (e.g. different cancer types), and thus the membership as well as the number of groups is known in advance, at least from a so-called *training data set*. This means that we also know the numbers of training set observations $n_1, \ldots, n_k$ for every group 1 to k, and they sum up to n. Moreover, for every observation we have measurements for p variables, thus vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, which are arranged in the rows of the $n \times p$ data matrix $\boldsymbol{X}$.

The goal of discriminant analysis is also very different from cluster analysis: Here we want to use the available information to establish *discriminant rules*, which later on will enable that new *test set observations* can be “discriminated”, thus assigned to the different groups. Assume a variable G which stands for the predicted group membership; thus, G will take on values from the set $\{1, \ldots, k\}$.

There are different approaches for discriminant analysis; here we will consider the **Bayesian approach**. For that we need to make further assumptions: We assume that the observations of the different groups are generated from multivariate normal distributions, with parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$, for $j = 1, \ldots, k$. Thus, this is similar to the assumption for model-based clustering, where we also had seen the density function $\boldsymbol{\phi}_j(\boldsymbol{x})$, see Equation (10.5). Also similar: We assume that the groups have *prior probabilities* p_j, where $p_1 + \ldots + p_k = 1$.

With that we can make use of the **Bayes theorem**:

$$\mathrm{P}(G = j|\boldsymbol{x}) = \frac{\boldsymbol{\phi}_j(\boldsymbol{x})p_j}{\sum_{l=1}^{k} \boldsymbol{\phi}_l(\boldsymbol{x})p_l} \tag{10.8}$$

The left-hand side is the *posterior probability*; it is a conditional probability: given an observation $\boldsymbol{x}$, what is the probability to have group label j. The right-hand side is simply the product of density and prior (as in model-based clustering!), divided by the sum of all these products. We would then assign the observation to that group for which we obtain the biggest value of the posterior probability.

If we just compare the posteriors of two groups, with index j and l, we could look at the (logarithm of the) ratio. Note that the denominator in Equation (10.8) is identical for all groups, and thus it cancels out in a ratio. This leads to

$$\log \frac{\mathrm{P}(G = j|\boldsymbol{x})}{\mathrm{P}(G = l|\boldsymbol{x})} = \log \frac{\boldsymbol{\phi}_j(\boldsymbol{x})p_j}{\boldsymbol{\phi}_l(\boldsymbol{x})p_l} = \log \frac{\boldsymbol{\phi}_j(\boldsymbol{x})}{\boldsymbol{\phi}_l(\boldsymbol{x})} + \log \frac{p_j}{p_l}. \tag{10.9}$$

If $\mathrm{P}(G = j|\boldsymbol{x}) > \mathrm{P}(G = l|\boldsymbol{x})$, the above log-ratio is positive, and observation $\boldsymbol{x}$ would be assigned to group j. This is already a classification rule – if we knew the value of $\boldsymbol{\phi}_j(\boldsymbol{x})$ and $\boldsymbol{\phi}_l(\boldsymbol{x})$. These depend on the yet unknown parameters of the normal distributions.

10.2.1 Linear discriminant analysis (LDA)

Let us make a further simplificaton by assuming $\boldsymbol{\Sigma}_1 = \ldots = \boldsymbol{\Sigma}_k$. Thus, all group covariances are supposed to be identical, and we call them $\boldsymbol{\Sigma}$. If we plug in $\boldsymbol{\Sigma}$ into the group densities in Equation (10.5), and consider the ratio of these densities in our rule (10.9), our trained mathematical eye immediately recognizes that some terms will vanish. This log-ratio finally translates to the so-called **linear discriminant rule:**

$\boldsymbol{x}$ is assigned to the j-th group (and not to the l-th group) if $\delta_j(\boldsymbol{x}) > \delta_l(\boldsymbol{x})$,

where

$$\delta_j(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \log p_j \tag{10.10}$$

is called *linear discriminant function* of the j-th group. The discriminant rule is indeed a linear function in $\boldsymbol{x}$, which justifies its name.

If the number of groups $k > 2$, we compute for the new test set observation $\boldsymbol{x}$ the linear discriminant function for every group, and $\boldsymbol{x}$ is assigned to that group for which we obtain the biggest values of the linear discriminant functions.

In order to be able to apply the rule (10.10), we first need to estimate the unknown parameters from the training data. The prior probabilities p_j can be estimated by the group proportions n_j/n if the training data are representative for the population. Let I_j be the index set of the observations of the j-th group of the training data. For estimating $\boldsymbol{\mu}_j$ we could use the arithmetic mean (vector) of the j-th group of the training data,

$$\hat{\boldsymbol{\mu}}_j = \bar{\boldsymbol{x}}_j = \frac{1}{n_j}\sum_{i\in I_j}\boldsymbol{x}_i \quad \text{for } j = 1,\ldots,k, \tag{10.11}$$

see also Equation (10.1). The joint covariance can be estimated by a "pooled" covariance,

$$\hat{\boldsymbol{\Sigma}} = \boldsymbol{S}_{pooled} = \frac{1}{n-k}\sum_{j=1}^{k}\sum_{i\in I_j}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)^T. \tag{10.12}$$

10.2.2 Quadratic discriminant analysis (QDA)

If we do not want to assume equal group covariances, then we will see that plugging in (10.5) into (10.9) will lead to a more complex expression. We end up with the so-called *quadratic discriminant functions*

$$\delta_j^{(q)}(\boldsymbol{x}) = -\frac{1}{2}\log(\det(\boldsymbol{\Sigma}_j)) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j) + \log p_j \tag{10.13}$$

for $j = 1,\ldots,k$, which are indeed quadratic in $\boldsymbol{x}$. A new observation $\boldsymbol{x}$ will then be assigned to that group for which we obtain the biggest value of the quadratic discriminant function.

As before we first need to estimate the unknown parameters by using the training data. In contrast to LDA we now do not need a joint covariance estimation, but we can simply use individual covariance estimates, e.g. by taking the sample covariance matrices

$$\hat{\boldsymbol{\Sigma}}_j = \boldsymbol{S}_j = \frac{1}{n_j - 1}\sum_{i\in I_j}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)^T. \tag{10.14}$$

Also more robust estimators could be used, such as the covariance estimation from the MCD estimator. Note that for QDA we need to estimate many more parameters than in LDA.

This could lead to an overfit of the discriminant rule to the training data, with the risk of a poorer performance for the test data.

As an example we again consider the iris data set. In contrast to cluster analysis we now make use of the grouping information to estimate the discriminant functions. If we would evaluate the perfomance on the same "training data" (and we just have a single data set), we would get a "too optimistic" impression of the classifier. This can be avoided by first randomly splitting the data into training and test set (which could be done several times).

LDA (QDA) can be computed in R as follows:

```
# R code LDA (QDA)
data(iris)                        # iris data
X <- iris[,1:4]                   # measurements
grp <- iris[,5]                   # grouping information
grpn <- as.numeric(grp)           # group numbers
set.seed(123)                     # to obtain same random selection
n <- nrow(iris)                   # number of observations
train <- sample(n,round(n*2/3)) # indexes of training data
test <- (1:n)[-train]             # indexes of test data
library(MASS)
res <- lda(X[train,],grp[train])   # LDA for training data
### for QDA simply the function qda()
res.pred <- predict(res,X[test,])  # prediction for test data
table(grp[test],res.pred$class)    # comparison with true grouping
#             setosa versicolor virginica
#  setosa         14          0         0
#  versicolor      0         17         2
#  virginica       0          0        17
#
plot(res,abbrev=2,col=grpn[train])       # linear discriminant functions
points(res.pred$x,col=grpn[test],pch=as.numeric(res.pred$class)) # test data
```

Figure 10.6 shows a projection of the data into the space of the linear discriminant functions. The textual symbols are referring to the training data, and color corresponds to the groups. The test data set is represented by non-textual symbols; color corresponds to the true group membership, symbol to the predicted one. We can see that only two (red) test set observations are incorrectly predicted.

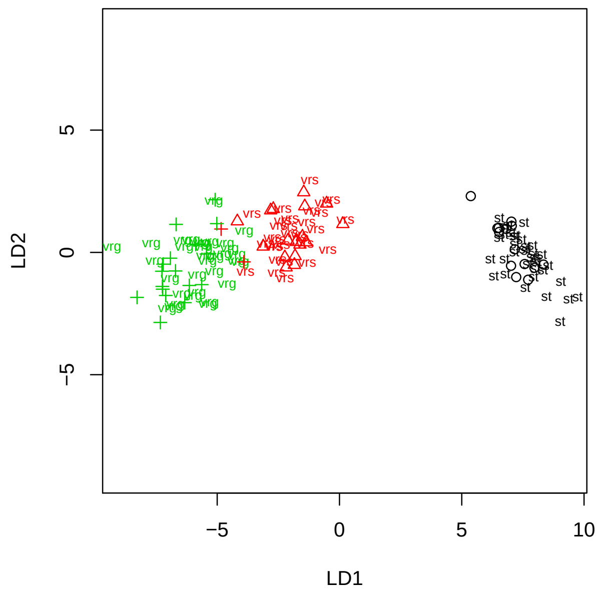


Figure 10.6: Results of LDA for the iris data set.

Chapter 11

Sampling design: basic considerations

11.1 Introduction

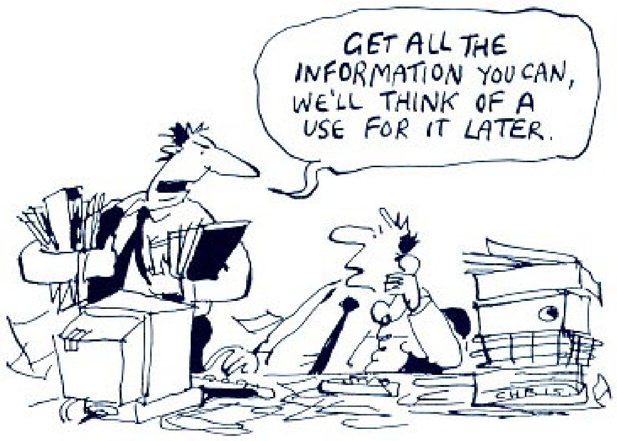

GET ALL THE INFORMATION YOU CAN, WE'LL THINK OF A USE FOR IT LATER.

Figure 11.1: Data, data, data, data, ...

This motto would not be useful for meaningful statistical conclusions, because:

- Collecting information can be cost and time intensive. Collecting a lot of information can be even more costly and takes more time.
- Already when collecting information we should have an idea about the problem setting and even about the possible statistical approach (result-oriented data collection).
- Also here we have: quality rather than quantity
- This is precisely the goal of statistics: also with fewer information we should be able to come up with conclusions (predictions).
- Data need to be comparable if they should be analyzed jointly. In the course of time the general framework could change, which makes the information incomparable.
- Statistics is different from detective work, where we would search for "any" hints. In statistics we want to come up with general conclusions based on representative observations.

11.2 Concepts

Population: The **population** is the collective of all statistical units with coherent identification criteria. Examples for statistical units are persons, households, or occurances. Examples for populations are all persons with main residence in Austria at a certain reference date.

Sample: A sample is a subset of a population within a statistical survey. The sample should be composed in a way to investigate certain properties of the whole population. If we are interested in the average household income in Austria, the sample will consist of a subset of all possible Austrian households.

Based on the sample we want to draw conclusions on the population. This is done because usually it is impossible to get detailed information from the whole population, or because it would be too expensive or too time consuming to "ask" (measure) all units of a population.

Sample design: The sample design refers to the selection criterion which is used to take a sample from the population. The first principle is to obtain a sample which is *representative* in order to draw valid conclusions from the analysis for the population.

Representative sample: A representative sample needs to reflect the complexity and the composition of the population. If we are interested in the mean household income, the sample needs to consider the different characteristics (age, gender, citicenship, income component, etc.) of the structure of a population. Figure 11.2 shows this schematically with a *mosaic plot* (the number of samples in each category is proportional to the represented area): the structure of the population (left) and that of the sample (right) are in good agreement.

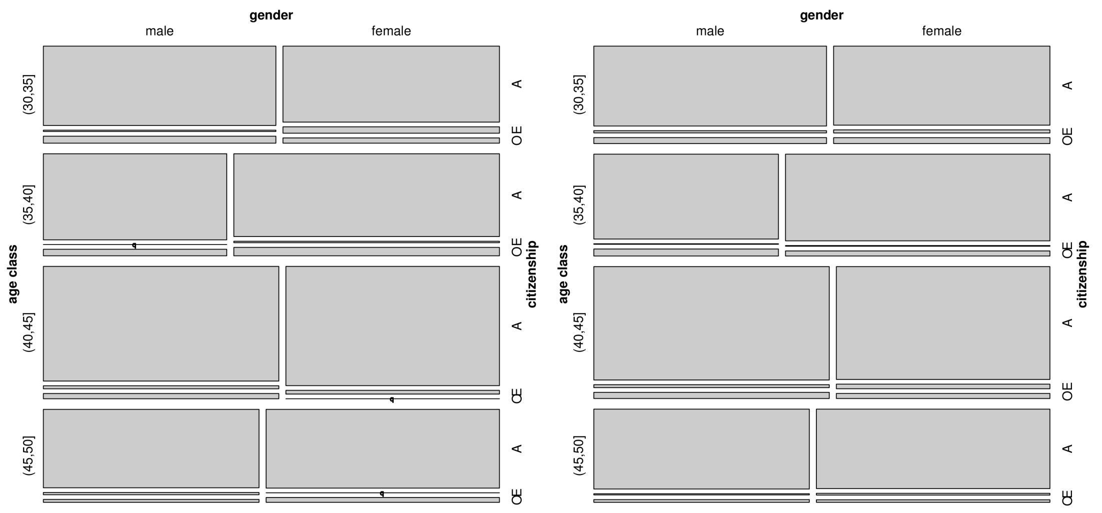


Figure 11.2: Population left, sample right.

Random sample versus quota sample: This refers to two strategies (sampling designs) to obtain a sample. For a random sample we select observations randomly; for a quota sample we proceed according to a specific scheme in order to have different groups represented in the sample. The selection according to Figure 11.2 can only be based on a quota sample, because with a random selection the different categories would not correspond so precisely to the population. However, within a quota sample, the samples in a group are selected randomly.

11.3 Planning the data collection

First we need to define the problem setting and the aim of the analysis. For answering the research questions we need to decide on:

- How to define the *population*?
- Which *statistical units* are used for data collection?
- Which *variables* should be analyzed?
- How should we *measure* the information (counts, index, etc.)?
- How much information should we collect (usually defining the costs)?

Example: An online company wants to improve the effectiveness of advertising their products. The population consists of all computers (IP addresses) which accessed their products. The statistical units to be measured could consist of all accesses from IP addresses to selected products in a certain time interval. The variables could be the number of sales per time interval of these products. If personal data to the IP addresses are known, we could collect further information such as age, gender, residential district, etc. As values of the variables we obtain count data or categories, as well as numbers and categories for age, gender, residential district, which can directly be processed. The type of the survey is internet-based (and not based on questionnaires), and the amount is defined by the selected time interval and products.

11.4 Statistical data acquisition

We distinguish among three different types of data:

Primary data collection: We ourselves are collecting the data. This can be done by questionnaires, by measuring, recording, etc. We need to consider all previously mentioned issues, starting from an appropriate planning, up to the selection of a representative sample. Especially with surveys we need to make sure that personal information is not accessible, and that manipulation by the interviewer ("leading questions") has to be strictly avoided.

Secondary data collection: We make use of existing data bases, such as data bases from Statistics Austria or EuroStat, data bases from banks, insurance companies, other companies, etc. A disadvantage of such data bases is that they possibly do not match with the context of the research question. They could also be outdated, or they could have poor data quality.

Tertiary data collection: We make use of existing aggregated data. For example, sales revenues of companies are usually only accessible in an aggregated form, where the aggregation is according to spatial aspects (district, etc.) or according to industrial sectors (or both).

11.5 Statistical data preparation

After having access to data, we are still not ready for the analysis. Usually we first need to prepare and preprocess the data for the analysis. Some common problems are:

Encoding: We have encoded data (categories, etc.), but not numbers, which would be needed for the analysis. A numerical encoding is not always necessary or useful, because in some cases we can also work with factor variables (e.g. for discriminant analysis, or simply for boxplot comparisons).

Data cleansing The data set could contain values which are not plausible, and thus they need to be checked and probably corrected. Data could contain *outliers*, which can either be corrected, or if this is not meaningful, we need to reduce their influence by applying robust statistical procedures. *Missing values* are a problem for many statistical methods. If a replacement of missings is not possible (by a certain statistical procedure), we need to use methods which can handle missings. For some statistical methods it might be necessary to do *data transformations* first.

11.6 Statistical data analysis

There are many different statistical software environments available, such as the open source software `R` (`http://cran.r-project.org`), or commercial products such as SAS, Statistica, SPSS, Stata, Eviews, Minitab.

It is important to understand what the functions actually do, and how the output should be interpreted. Therefore, a deeper mathematical understanding of the methods is always useful and recommended.

11.7 Interpretation of results

The interpretation of the results requires a deeper understanding of the methods. Especially the statistical assumptions of the methods need to be considered for the validity of the results. More general content-wise interpretations should be done by the domain experts, ideally with support from the data analysts.

..KANN ICH TROTZ DER ANFÄNGLICHEN POLEMIK ZU MEINER QUALIFIKATION DIESES GRANDIOSE ERGEBNIS PRÄSENTIEREN!
SIE HALTEN DAS BLATT FALSCH RUM!
x
y
M2011