

# Statistical Computing

## Introduction to

---

Günther Sawitzki  
StatLab Heidelberg  
*private Version 16. Februar 2008*

*in preparation*



---

# Contents

---

<b>Introduction</b>	<b>9</b>
<b>1 Basic Data Analysis</b>	<b>1-1</b>
1.1 R Programming Conventions	1-1
1.2 Generation of Random Numbers and Patterns	1-4
1.2.1 Random Numbers	1-4
1.2.2 Patterns	1-9
1.3 Case Study: Distribution Diagnostics	1-10
1.3.1 Distribution Functions	1-12
1.3.2 Histogram and Related Plots	1-16
Bar Charts	1-20
1.3.3 Statistics of Distribution Functions; Kolmogorov-Smirnov-Tests	1-22
Monte Carlo Confidence Bands	1-23
1.3.4 Statistics of Histograms and Related Plots; $\chi^2$ -Tests	1-29
1.4 Moments and Quantiles	1-35
1.5 R Complements	1-39
1.5.1 Random Numbers	1-39
1.5.2 Graphical Comparisons	1-40
1.5.3 Enhancing Graphical Displays	1-46
1.5.4 Functions	1-47
1.5.5 R Internals	1-51
Parse	1-51
Eval	1-51
Print	1-52

Executing Files	1-52
1.5.6 Packages	1-52
1.6 Statistical Summary	1-55
1.7 Literature and Additional References	1-55
<b>2 Regression</b>	<b>2-1</b>
2.1 General Regression Model	2-1
2.2 Linear Model	2-2
2.2.1 Factors	2-4
2.2.2 Least Squares Estimation	2-5
2.2.3 More Examples for Linear Models	2-15
2.2.4 Model Formulae	2-16
2.2.5 Gauss-Markov Estimator	2-17
2.3 Variance Decomposition and Analysis of Variance	2-19
2.4 Simultaneous Inference	2-26
2.4.1 Scheffé's Confidence Bands	2-26
2.4.2 Tukey's Konfidence Intervals	2-28
Case Study: Microplates	2-29
2.5 Beyond Linear Regression	2-35
Transformations	2-35
Box-Cox-Transformations	2-35
2.5.1 Generalized Linear Models	2-35
2.5.2 Local Regression	2-36
2.6 R Complements	2-40
2.6.1 Discretisation	2-40
2.6.2 External Data	2-40
2.6.3 Testing Software	2-40
2.6.4 R Data Types	2-41
2.6.5 Classes and Polymorphic Functions	2-42
2.6.6 Extraktor Functions	2-43
2.7 Statistical Summary	2-44
2.8 Literature and Additional References	2-44

CONTENTS	5
<b>3 Comparison of Distributions</b>	<b>3-1</b>
3.1 Shift/Scale Families	3-4
3.2 <i>QQ</i> -Plot, <i>PP</i> -Plot, and Comparison of Distributions	3-6
3.2.1 Kolmogorov Smirnov Tests	3-11
3.3 Tests for Shift Alternatives	3-12
3.4 Power and Confidence	3-20
3.4.1 Theoretical Power and Confidence	3-20
3.4.2 Simulated Power and Confidence	3-24
3.4.3 Quantilschätzung durch Simulation	3-27
3.5 Qualitative Features of Distributions	3-29
3.6 R Complements	3-31
3.7 Statistical Summary	3-33
3.8 Literature and Additional References	3-33
<b>4 Dimensions 1, 2, 3, ..., <math>\infty</math></b>	<b>4-1</b>
4.1 R Complements	4-1
4.2 Dimensions	4-4
4.3 Selections	4-5
4.4 Projections	4-6
4.4.1 Marginal Distributions and Scatterplot Matrices	4-7
4.4.2 Projection Pursuit	4-12
4.4.3 Projections for Dimensions 1, 2, 3, ... 7	4-14
4.4.4 Parallel Coordinates	4-14
4.5 Sections, Conditional Distributions and Coplots	4-16
4.6 Transformations and Dimension Reduktion	4-22
4.7 Higher Dimensions	4-27
4.7.1 Linear Case	4-27
Partial Residuals and Added Variable Plots	4-28
4.7.2 Nonlinear Case	4-29
Example: Cusp Nonlinearity	4-29
4.7.3 “Curse of Dimension”	4-34
4.7.4 Case Study	4-34
4.8 High Dimensions	4-46
4.9 Statistical Summary	4-48
4.10 Literature and Additional References	4-48

<b>R as a Programming Language and Environment</b>	<b>A-49</b>
A.11 Help and Information	A-49
A.12 Names and Search Paths	A-49
A.13 Customizing	A-51
A.14 Basic Data Types	A-53
A.15 Output for Objects	A-55
A.16 Object Inspection	A-57
A.17 System Inspection	A-59
A.18 Complex Data Types	A-61
A.19 Accessing Components	A-63
A.20 Data Manipulation	A-65
A.21 Operators	A-67
A.22 Functions	A-69
A.23 Debugging and Profiling	A-71
A.24 Control Structures	A-73
A.25 Administration and Customisation	A-75
A.26 Input and Output to Data Streams	A-77
A.27 External Data	A-79
A.28 Libraries, Packages	A-81
A.29 Linear Algebra	A-83
A.30 Model Descriptions	A-85
A.31 Graphic Functions	A-87
A.31.1 High Level Graphics	A-87
A.31.2 Low Level Graphics	A-87
A.31.3 Annotations and Legends	A-88
A.31.4 Graphic Parameters and Layout	A-89
A.32 Basic Statistical Functions	A-91
A.33 Distributions, Random Numbers, Densities...	A-93
A.34 Programming on the Language	A-95
<b>Bibliography</b>	<b>A-97</b>

CONTENTS	7
<b>Functions by Topic</b>	<b>A-99</b>
<b>Function and Variable Index</b>	<b>A-100</b>
<b>Index</b>	<b>A-104</b>
<b>B</b>	<b>B-1</b>
B.1 Test R function index	B-1
B.2 R styles	B-2
B.3 Tables	B-5
<b>C S help — remove for public version</b>	<b>C-1</b>
C.1 Ch 01	C-1
<b>D To Do</b>	<b>D-1</b>
D.1 For release	D-1
D.2 misc	D-1



---

# Introduction

---

ch:00

This introduction to R is intended as course material to be used in a compact course or for self-instruction. The course is for students with basic knowledge in stochastics. Notions like distribution function, quantile, expected value and variance are presupposed, as well as some familiarity with statistical features corresponding to these notions. Classical distributions, such as binomial, uniform, and gaussian should be familiar, together with derived distributions and their asymptotic behavior. It is not required that the reader has knowledge in statistics proper. But this knowledge is not taught in this course, on the other hand. This course concentrates on the “computing” aspects. Statistical concepts and statistical points of view are introduced and discussed. But for an in depth discussion, the reader is referred to statistics courses,

A working knowledge in computer usage is presupposed, at least rudimentary knowledge of programming concepts like variables, loops, functions. No extended knowledge in computing is required.

s:01

## What is R?

R is a programming language, and the name of a software system which implements this language. The R programming language has been developed for statistics and stochastic simulation. By now, it has become a standard in these fields. To be precise, we should use specific terms: the language is called S, its implementation and the environment are called R. The original authors of S are John M. Chambers, R. A. Becker and A. R. Wilks, AT & T Bell Laboratories, Statistics Research Department. The language and its development are documented in a series of books, commonly called by their cover as white ([3]), blue ([1]) and green book ([2]).

For a long time the AT & T implementation of S has been the reference for the S language. Today, S is available in a commercial version called S-Plus <<http://www.insightful.com/>> (based on the AT & T implementation) and as a free software version R\*, or “Gnu S” <<http://www.r-project.org/>>.



\* R got its name by accident - the same accident which made the first names of the original authors of R (Ross Ihaka & Robert Gentleman) start with R.

In the mean time, R has become the reference implementation. Essential precisions and - if necessary - even modifications of the language are defined by R. For simplicity, here and in the sequel we use “R language” as a common term even where the precise term should be: the S language using the R implementation.

R is an interpreted programming language. Instructions in R are executed immediately. Besides the original elements of S, R has several extensions. Part of these are introduced in response to recent developments in statistics, part are introduced to open experimental facilities. On the other side, advancements of the S language are taken into account.

The recent (2008) version of R is 2.x. This version is largely compatible with the previous version R 1.x. The essential changes are internal to the system. For initial use, there is no significant difference from R 1.x. For the advanced user, there are three essential innovations:

- *Graphics*: The basic graphic system in R implements a models inspired by pen and paper drawing. A graphics port (paper) is opened, and lines, points or symbols are drawn in this port. In R 2.x there is a second additional graphics system, oriented at a camera/object model. Graphical object in various position and orientation are mapped in a visual space.
- *Packages*: The original R had a linear command history and a uniform work space. R 2.x introduced an improved support for “packages” which may have encapsulated information. To support this, language concepts such as “name spaces” and various tools have been introduced.
- *Internationalization*: Originally, R was based on the English language, and ASCII was the general encoding used. With R 2.x extensive support for other languages and encodings has been introduced. With this, it has become possible to provide localized versions.

Two aspects of R are active areas of recent developments: interactive access, and integration in a networked environment. These and other aspects are part of Omegahat - an attempt to develop a next generation system based on the experiences from R . These more experimental work is accessible at <<http://www.omegahat.org/>>. R does already provide simple possibilities to call functions implemented in other languages like C or FORTRAN. Omegahat extends these possibilities and allows direct access to Java, Perl .... A Java based graphical interface for R is JGR, accessible at <<http://stats.math.uni-augsburg.de/software/>>. A collection of interactive displays for R is in *ipplots*, available at the same site.

Recent developments related to R are in <<http://r-forge.r-project.org/>>. Many helpful extensions are in <<http://www.bioconductor.org/>>, a site which is targeted at biocomputing.

s:02

R has been developed for practical work in statistics. Usability often has been given priority over abstract design principles. As a consequence, it is not easy to give a systematic introduction to R. A devoured path is chosen instead: case studies and examples, followed by systematic surveys. Practical work should make use of the rich online material which comes with R. Starting points are the “frequently asked questions” (FAQ)

<<http://www.cran.r-project.org/faqs.html>>. “An Introduction to R” ([14]) is the “official” introduction. This documentation and other manuals can be downloaded from <<http://www.cran.r-project.org/manuals.html>>.

Many R functions come with the basic system. Other functions must be loaded from libraries. Some of these libraries come with the R system. If additional packages are needed, <<http://www.cran.r-project.org/>> is a first resource for downloads.

The commercial S-Plus comes in various versions. S-Plus 4.x and S-Plus 2000 use S version 3 and are largely compatible with R. S-Plus 5 is an implementation of S version 4 with some changes which need attention. These programming details of S-Plus are not discussed here. Information about S-Plus is at <<http://www.insightful.com/>>.

### s:03 Content and Outline of this Course

In its basic version, R contains more than 1500 functions - too many to introduce in just one course, and too many to learn reasonably. The course can only open the way to R.

Course participants can come from various background, with different prerequisites. For pupils and younger students, a mere programming course on technical basics may be appropriate. Later in study, questions about meaningful classification and background will be more important. This is what the present course aims at. The “technical” material gives a skeleton. Beyond this, we try to open the view for statistical questions and rise interest for the background. The course should sharpen the appetite for the substance which may be offered in a subsequent well-founded statistical course.

The first part of this course material is organized by themes, using example topics to illustrate how R can be used to tackle statistical problems.

An appendix gives a collection of R language elements and functions. During the course, it can help es a quick reference and at may serve as a starting point and orientation to access the rich information material which comes bundled with R. After the course, it may serve as a note pad. Finally, on the long run for practical work, the online help and online manuals of R are the prime sources of information.

Using a selection of the exercises, the course can be completed within about four days of work. Conceptually, it is a four day introduction to statistics with the topic areas

- One sample analysis and distributions
- Regression
- Two sample problems and comparison of distributions
- Multivariate analysis

A generous time slot for exercises is recommended (an additional half day for the introductory exercises, an additional half day for one of the project exercises). With this, the course can be covered in one week, provided follow up facilities are established to answer questions which have come up, and possibilities are available to follow the interest in statistical background which may have resulted.

**ToDo:** examples of help files included

**ToDo:**

comment  
on litera-  
ture

**ToDo:**

comment  
ch.4 expe-  
rimental

expl:00

Using the course during a term in a weekly class needs more time, since repetitions must be calculated in. Each of the first four chapters will cover about four lectures, plus time for exercises. For this time schedule, a course to cover the statistical background is recommended running in parallel.

For a subsequent self-paced study going into details on R as a programming language, the recommended reading is ([24]).

### Typographical Conventions

Examples and input code are formatted so that they can be used with “Cut & Paste” and entered as program input. To allow this, punctuation marks are omitted and the input code is shown without a “prompt”. For example

*Example 0.1:*

1 + 2

*Input*

3

*Output*

may correspond to a screen output of

```
> 1+2
[1] 3
>
```

Depending on the configuration, the prompt ">" may be represented by a different Symbol.

### Thanks

Thanks to the R core team for comments and hints. Special thanks to Friedrich Leisch (R core team) and Antony Unwin (Univ. Augsburg).

### Literature and Additional References

s:04

R:Rnotes

[14] R Development Core Team (2000-2007): An Introduction to R.  
See: <<http://www.r-project.org/manuals.html>>.

**R:Rref**  
[17] R Development Core Team (2000-2007): R Reference Manual.  
See: <<http://www.r-project.org/manuals.html>>.

The Omega Development Group (2000): Omega.  
See: <<http://www.omegahat.org/>>.

**bcw88news**  
[1] Becker, R.A.; Chambers, J.M.; Wilks, A.R. (1988): The New S Language. New York:  
Chapman and Hall.

**chh92sns**  
[3] Chambers, J.M.; Hastie, T.J. (eds) (1992): Statistical Models in S. New York: Chapman  
and Hall.

**clev93vd**  
[4] Cleveland, W.F. (1993): Visualizing Data. Summit: Hobart Press.

**vr02mass**  
[25] Venables, W.N.; Ripley, B.D. (2002): Modern Applied Statistics with S. Heidelberg:Springer.  
See: <<http://www.stats.ox.ac.uk/pub/MASS4/>>.

**vr00pis**  
[24] Venables, W.N.; Ripley, B.D. (2000): Programming in S. Heidelberg:Springer.  
See: <<http://www.stats.ox.ac.uk/pub/MASS3/Sprog>>.

```
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/chapters/S00eintro.tex,v $  
$Revision: 1.1 $  
$Date: 2008/01/26 20:27:31 $  
$Name: $  
$Author: j40 $
```

---

## CHAPTER 1

---

# Basic Data Analysis

---

ch:01

### 1.1 R Programming Conventions

s:1.0

Like any programming language, R has certain conventions. Here are the first basic rules.

**To Do:**

discuss

missing  
values

-

IntroR 2.5

Rconventions	
numbers	A point is used as a decimal separator. Numbers can be written in exponential form; the exponential part is introduced by <i>E</i> . Numbers can be complex numbers; the imaginary part is marked by <i>i</i> . <i>Beispiel:</i> 1 2.3 3.4E5 6i+7.8  Numbers can take the values <i>Inf</i> , <i>-Inf</i> , <i>NaN</i> for “not a number” and <i>NA</i> for “not available” = missing. <i>Beispiel:</i> 1/0 results in <i>Inf</i> ; 0/0 results in <i>NaN</i> .
Strings	Strings are delimited by " or '. <i>Beispiel:</i> "ABC" 'def' "gh'ij"

To allow for non trivial examples, we anticipate a detail: in R, *a:b* is a sequence of numbers from *a* to at most *b* in steps of 1 resp. -1.

R conventions	
	(Fortsetzung)→

<b>R conventions</b> (Fortsetzung)	
Objects	The basic elements in R are objects. Objects can be assigned to classes. <i>Beispiel:</i> The basis objects in R are vectors.
Names	R objects can have names, by which they can be accessed. Names begin with a letter or a dot, followed by a sequence of letters, digits, or the special characters _ or . <i>Beispiele:</i> <code>x</code> <code>y_1</code> Lower and upper case are treated as different. <i>Beispiele:</i> <code>y87</code> <code>y87</code>
Assignments	Assignments have the form <i>Aufruf:</i> <code>name &lt;- value</code> or alternatively <code>name = value</code> . <i>Beispiel:</i> <code>a &lt;- 10</code> <code>x &lt;- 1:10</code>
Queries	If only the name of an object is entered, the value of the object is returned. <i>Beispiel:</i> <code>x</code>
Indices	Vector components are accessed by index. The lowest index is 1. <i>Beispiel:</i> <code>x[3]</code> The indices can be specified directly, or using symbolic names or rules. <i>Beispiele:</i> <code>x[-3]</code> <code>x[ x^2 &lt; 10 ]</code> <code>a[1]</code>

<b>help and inspection</b>	
----------------------------	--

(Fortsetzung)→

<i>help and inspection</i> (Fortsetzung)	
help	<p>Documentation and additional information about an object can be requested using <code>help</code>.</p> <p><i>Aufruf:</i> <code>help(name)</code></p> <p><i>Beispiele:</i> <code>help(exp)</code> <code>help(x)</code></p> <p>Alternative form <code>?name</code></p> <p><i>Beispiele:</i> <code>?exp</code> <code>?x</code></p>
Inspection	<p><code>help()</code> can only provide information that has been prepared in advance. <code>str()</code> can inspect the actual state of an object and display this information.</p> <p><i>Aufruf:</i> <code>str(object, ...)</code></p> <p><i>Beispiele:</i> <code>str(x)</code></p>

<i>R conventions</i>	
Functions	<p>Function calls in R have the form</p> <p><i>Aufruf:</i> <code>name(parameter ... )</code></p> <p><i>Beispiel:</i> <code>e_10 &lt;- exp(10)</code></p> <p>This convention even hold when there are no parameters at all.</p> <p><i>Beispiel:</i> To quit R, you call a “quit” function <code>q()</code>.</p> <p>Parameters are treated in a very flexible way. They can have default values, which are used if no explicit parameter value is given.</p> <p><i>Beispiele:</i> <code>log(x, base = exp(1))</code></p> <p>Functions can be <b>polymorphic</b> sein. For a polymorphic function, the actual function is determined by the class of the actual parameters.</p> <p><i>Beispiele:</i> <code>plot(x)</code> <code>plot(x, x^2)</code> <code>summary(x)</code></p>

(Fortsetzung)→

<pre>random_numbers runif Uniform@Unifor dunif@dunif   (Uni-   form) punif@punif   (Uni-   form) qunif@qunif   (Uni-   form) runif@runif   (Uni-   form) Topic dis- tri- bu- tion!Uniform@Uniform</pre>	<p><b>R conventions</b> (Fortsetzung)</p> <p>Operators</p> <p>When applied to vectors, operators operate on each of the vector components.</p> <p><i>Beispiel:</i> For vectors <math>y</math>, <math>z</math>, the product <math>y*z</math> is the vector of componentwise products.</p> <p>Operators are special functions. They can be called in prefix form (function form).</p> <p><i>Beispiel:</i> <math>+(x, y)</math></p> <p>When applied to two operands with different lengths, the smaller operand is repeated cyclically.</p> <p><i>Beispiel:</i> <math>(1:2)*(1:6)</math></p>
---	---

We beschäftigen uns im folgenden mit statistischen Methoden. Wir benutzen die Methoden zunächst in Simulationen, d.h. mit synthetischen Daten, deren Erzeugung wir weitgehend unter Kontrolle haben. Das erlaubt es uns, Erfahrung mit den Methoden zu gewinnen und sie kritisch zu beurteilen. Erst dann benutzen wir die Methoden zur Analyse von Daten.

## 1.2 Generation of Random Numbers and Patterns

s:1.1

### 1.2.1 Random Numbers

The function `runif()` erlaubt die Erzeugung von uniform verteilten random variables. Mit `help(runif)` oder `?runif` erhalten wir Informationen, wie die function benutzt werden kann:

**help(runif)**

**Uniform**

*The Uniform Distribution*

#### Description

These functions provide information about the uniform distribution on the interval from `min` to `max`. `dunif` gives the density, `punif` gives the distribution function `qunif` gives the quantile function and `runif` generates random deviates.

*Usage*

```
dunif(x, min=0, max=1, log = FALSE)
punif(q, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
runif(n, min=0, max=1)
```

```
.Random.seed@.Random.seed
rnorm@rnorm|textit
```

*Arguments*

<code>x, q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>n</code>	number of observations. If <code>length(n) &gt; 1</code> , the length is taken to be the number required.
<code>min, max</code>	lower and upper limits of the distribution.
<code>log, log.p</code>	logical; if TRUE, probabilities p are given as log(p).
<code>lower.tail</code>	logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ .

*Details*

If `min` or `max` are not specified they assume the default values of 0 and 1 respectively.  
The uniform distribution has density

$$f(x) = \frac{1}{max - min}$$

for  $min \leq x \leq max$ .

For the case of  $u := min == max$ , the limit case of  $X \equiv u$  is assumed.

*References*

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

*See Also*

`.Random.seed` about random number generation, `rnorm`, etc for other distributions.

*Examples*

```
u <- runif(20)

## The following relations always hold :
punif(u) == u
dunif(u) == 1

var(runif(10000))#- ~ = 1/12 = .08333
```

Serienplot|textbf  
plot

---

Diese Hilfsinformation sagt uns: Als parameter for `runif()` muss die Anzahl `n` der zu generierenden Zufallsvalues angegeben werden. Als weitere parameter for `runif()` können das Minimum und das Maximum des valuesbereichs angeben werden. Geben we keine weiteren parameter an, so werden die Default-values `min = 0` and `max = 1` genommen. Z. B. `runif(100)` erzeugt einen vector with 100 uniform verteilten random variables im Bereich (0, 1). Der Aufruf `runif(100, -10, 10)` erzeugt einen vector with 100 uniform verteilten random variables im Bereich (-10, 10). The zusätzlichen parameter können in der definierten Reihenfolge angegeben werden, or mithilfe der names spezifiziert werden. Bei Angabe des name kann die Reihenfolge frei gewählt werden. Anstelle von `runif(100, -10, 10)` kann also `runif(100, min = -10, max = 10)` or `runif(100, max = 10, min = -10)` benutzt werden. Dabei können auch ausgewählt einzelne parameter gesetzt werden. Wird zum Beispiel das Minimum nicht angegeben, so wird for das Minimum der Default-value eingesetzt: die Angabe von `runif(100, max = 10)` is gleichwertig with `runif(100, min = 0, max = 10)`. Der besseren Lesbarkeit halber geben we oft die names von Parametern an, auch falls es nicht nötig is.

Jeder Aufruf von `runif()` erzeugt 100 neue uniforme random numbers. We können diese speichern.

```
x <- runif(100)
```

erzeugt einen neuen vector von random numbers and weist ihn der Variablen `x` zu.

```
x
```

gibt jetzt dessen values aus; damit können we die Resultate inspizieren. Eine grafische Darstellung, den **Serienplot** - einen Scatterplot der Einträge in `x` gegen den laufenden Index, erhalten we with

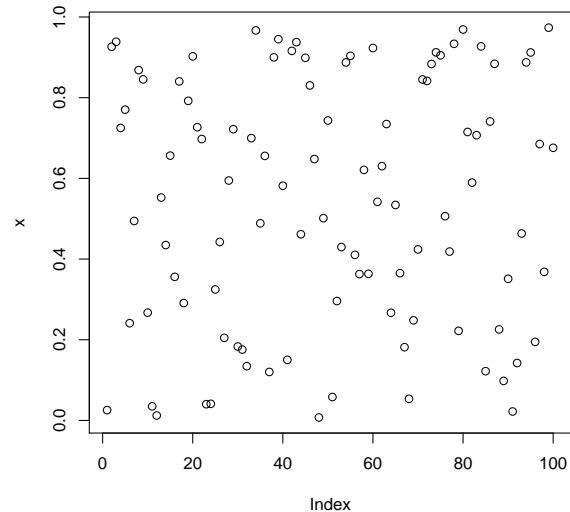
```
plot(x)
```

expl:ch01-Sframe02

**Example 1.1:**

*Input*

```
x <- runif(100)
plot(x)
```



a:ch01:01-1

Aufgabe 1.1	
	<p>Experimentieren Sie with den Plots and <code>runif()</code>. Sind die Plots Bilder von random numbers?</p> <p>Genauer: Akzeptieren Sie die Plots als Bilder von 100 unabhängigen Realisationen von uniform auf (0, 1) verteilten random numbers?</p> <p>Wiederholen Sie das Experiment and versuchen Sie, die Argumente, die for or gegen die (uniforme) Zufälligkeit sprechen, möglichst genau zu notieren. Ihr Resumée?</p> <p>Gehen Sie die Argumente noch einmal durch and versuchen Sie, eine Prüfstrategie zu entwerfen, with der Sie eine sequence von numbers auf (uniforme) Zufälligkeit überprüfen könnten. Versuchen Sie, diese Strategie möglichst genau zu formulieren.</p> <p style="text-align: right;">(Fortsetzung)→</p>

Pseudo-  
random  
num-  
bers | see random  
numbers | text  
random\_numbers !Pseudo-

Aufgabe 1.1	(Fortsetzung)
	<p><i>Hinweis:</i> Sie können mehrere Abbildungen in a Fenster halten. Mit <code>par(mfrow = c(2, 3))</code> wird das Grafik-System so eingestellt, dass jeweils sechs Abbildungen zeilenweise als <math>2 \times 3</math>-Matrix angeordnet (2 Zeilen, 3 Spalten) gezeigt werden.</p> <p>The function <code>par</code> is die zentrale function, with der die Grafik-Ausgabe parametrisiert wird. Weitere Information erhält man with <code>help(par)</code>.</p>

**ToDo:**  
 consider  
 simple  
 exercises  
 for random  
 numbers –  
 Antony

We lüften gleich das Geheimnis\*: die numbers are nicht zufällig, sondern ganz deterministisch. Genauer: im Hintergrund von `runif()` wird eine deterministische sequence  $z_i$  generiert. Verschiedene Algorithmen stehen zur Verfügung. Informationen dazu erhält man with `help(.Random.seed)`. Im einfachsten Fall, for lineare Kongruenzgeneratoren, werden aufeinanderfolgende values  $z_i, z_{i+1}$  sogar nur with einer linearen function generiert. Damit die values im kontrollierten Bereich bleiben, wird modulo einer oberen Grenze gerechnet, also

$$z_{i+1} = a z_i + b \mod M.$$

The resultierenden values, die uns übergeben werden, are umskaliert auf

$$\frac{z_i}{M} \cdot (max - min) + min.$$

The dadurch definierte sequence kann regelmäßig sein and schnell zu periodischer Wiederholung führen. Bei geeigneter Wahl der parameter, wie beim Beispiel in der Fußnote, kann sie jedoch zu einer sehr langen Periode (in der Größenordnung von  $M$ ) führen and scheinbar zufällig sein. The numbersfolge is jedoch keine unabhängige Zufallsfolge, and die distribution is auch nicht uniform auf  $(min, max)$ .

Selbst wenn man das Geheimnis kennt, is es nur with viel weiterem Wissen möglich nachzuweisen, dass die erzeugte sequence nicht den Gesetzen folgt, die for eine unabhängige sequence von identisch uniform verteilten random numbers gelten.

numbersfolgen, die den Anspruch erheben, sich wie zufällige numbers zu verhalten, nennen we **Pseudo-random numbers**, wenn es wichtig is, auf den Unterschied hinzuweisen. We benutzen diese Pseudo-random numbers, um uns geeignete Test-Datensätze zu generieren. We können damit untersuchen, wie sich statistische Verfahren unter nahezu bekannten Bedingungen verhalten. Dabei benutzen we Pseudo-random numbers, als ob we random numbers hätten.

Pseudo-random numbers sollten we zum anderen als Herausforderung nehmen: Sind we in der Lage, sie als nicht unabhängige random numbers zu erkennen? Wenn we einen Unterschied erkennen, werden we versuchen, den Pseudo-Zufallsnumbersgenerator gegen

\* ... nur teilweise. The benutzten Zufallsgeneratoren in R are konfigurierbar and können wesentlich komplexer sein, als hier vorgestellt. for unsre Diskussion reicht jedoch hier die Familie der linearen Kongruenzgeneratoren. Sie können deren Verhalten in anderen Programmiersystemen nachvollziehen. The übliche Referenz is dabei der “minimal standard generator” with  $x_{i+1} = (x_i \times 7^5) \mod 2^{31} - 1$ .

einen besseren auszutauschen. Aber zunächst geht die Herausforderung an uns. Sind wir überhaupt in der Lage, z.B. eine with a linear Generator erzeugte deterministische sequence als nicht zufällig zu erkennen? Falls nicht: welche intellektuellen Konsequenzen ziehen wir daraus?

### 1.2.2 Patterns

Außer Pseudo-random numbers gibt es in R eine ganze Reihe von Möglichkeiten, regelmäßige Sequenzen zu generieren. Die in anderen Sprachen notwendigen Schleifen werden damit weitgehend vermieden. Hier eine erste Übersicht:

R Sequenzen	
:	Erzeugt ganzzahlige Sequenz von <i>Anfang</i> bis <i>Ende</i> . <i>Aufruf:</i> <code>Anfang:Ende</code> <i>Beispiele:</i> <code>1:10</code> <code>10:1</code>
<code>c()</code>	“combine”. Kombiniert Argumente zu einem neuen vector. <i>Aufruf:</i> <code>c(..., recursive = FALSE)</code> <i>Beispiele:</i> <code>c(1, 2, 3)</code> <code>c(x, y)</code>  Bezeichnet die Argumente zusammengesetzte Datentypen, so arbeitet die function rekursiv absteigend in die Daten hinab, wenn sie mit <code>recursive = TRUE</code> aufgerufen wird.
<code>seq()</code>	Erzeugt allgemeine Sequenzen. <i>Aufruf:</i> Siehe <code>help(seq)</code>
<code>rep()</code>	Wiederholt Argument. <i>Aufruf:</i> <code>rep(x, times, ...)</code> <i>Beispiele:</i> <code>rep(x, 3)</code> <code>rep(1:3, c(2, 3, 1))</code>

Dabei steht “...” für eine variable Liste von Argumenten. Wir werden diese Notation noch häufiger benutzen.

Aufgabe 1.2	
	<p>Generieren Sie with</p> $\text{plot}(\sin(1:100))$ <p>einen Plot with einer diskretisierten Sinusfunktion. (Falls Sie die Sinusfunktion nicht sofort erkennen, benutzen Sie <code>plot(sin(1:100), type = "l")</code>, um die Punkte zu verbinden. Benutzen Sie Ihre Strategie aus Aufgabe 1.1. Können Sie damit die Sinusfunktion als nicht zufällig erkennen?)</p>

The numbersreihe eines Datensatzes, wie z.B. die Ausgabe eines Zufallsnumbersgenerators hilft selten, zugrunde liegende Strukturen zu erkennen. Nur wenig helfen einfache, unspezifische grafische Darstellungen wie der Serienplot. Selbst bei klaren Mustern are diese Informationen selten aussagekräftig. Zielgerichtete Darstellungen are nötig, um Verteilungseigenschaften zu untersuchen.

### 1.3 Case Study: Distribution Diagnostics

s:1.2

We brauchen genauere Strategien, um Strukturen zu erkennen or deren Verletzung festzustellen. Wie diese Strategien aussehen können, skizzieren we am Beispiel der random numbers. We konzentrieren uns hier auf die Verteilungseigenschaft. Angenommen, die sequence besteht aus unabhängigen random numbers with einer gemeinsamen distribution. Wie überprüfen we, ob dies die uniforme distribution is? We ignorieren die mögliche Umskalierung auf (min, max) - dies is ein technisches Detail, das die Fragestellung nicht wesentlich tangiert. We betrachten  $\min = 0; \max = 1$ .

Aus Realisierungen von random variables können distributions nicht direkt abgelesen werden. Dies is unser kritisches Problem. We brauchen Kennzeichnungen der distributions, die we empirisch überprüfen können. We können zwar observations als Maße betrachten: for  $n$  observations  $X_1, \dots, x_n$  können we formal die empirische distribution  $P_n$  definieren als das Maß  $P_n = \sum(1/n)\delta_{x_i}$ , wobei  $\delta_{x_i}$  das Dirac-Maß an der Stelle  $X_i$  is. Also

$$P_n(A) = \#\{i : X_i \in A\}/n.$$

Aber leider is das empirische Maß  $P_n$  einer Beobachtungsreihe von unabhängigen observations with gemeinsamem Maß  $P$  im allgemeinen sehr von  $P$  verschieden. Einige Eigenschaften gehen unwiederbringlich verloren. Dazu gehören infinitesimale Eigenschaften: so is z.B.  $P_n$  immer auf endlich viele Punkte konzentriert. We brauchen Konstrukte, die anhand von Realisierungen von random variables bestimbar and with den entsprechenden Konstrukten von theoretischen distributions vergleichbar are. Eine Strategie is es, sich auf (empirisch handhabbare) Testmengen zu beschränken.

#### S01:ex01 Example 1.1 *distribution function*

Anstelle der distribution  $P$  betrachten we ihre distribution function  $F = F_P$  with

$$F(x) = P(X \leq x).$$

for eine empirische distribution  $P_n$  von  $n$  observations  $X_1, \dots, X_n$  is entsprechend die empirische distribution function

$$F_n(x) = \#\{i : X_i \leq x\}/n.$$

S01:ex02

### Example 1.2 Histogramm

We wählen disjunkte Testmengen  $A_j, j = 1, \dots, J$ , die den valuesbereich von  $X$  überdecken. for die uniforme distribution auf  $(0, 1)$  können we z.B. die Intervalle

$$A_j = \left( \frac{j-1}{J}, \frac{j}{J} \right]$$

als Testmengen wählen.

Anstelle der distribution  $P$  betrachten we den vector  $(P(A_j))_{j=1, \dots, J}$  resp. den empirischen vector  $(P_n(A_j))_{j=1, \dots, J}$ .

We diskutieren diese examples ausführlicher. Einige allgemeine Lehren können we daraus ziehen. We machen mehrere Durchgänge, um von a naiven Zugang zu a entwickelten statistischen Ansatz zu kommen.

An dieser Stelle sei schon darauf hingewiesen, dass Histogramme kritisch von der Wahl der Testmengen abhängen. Insbesondere wenn Diskretisierungen in den Daten unglücklich with der Wahl der Testmengen zusammentreffen, kann es zu sehr irreführenden Ergebnissen kommen. Eine Alternative zu Histogrammen is es, die Daten zu glätten.

S01:kdens

**Example 1.3 Glättung** We ersetzen jeden Datenpunkt durch eine (lokale) distribution, d.h. we verschmieren die Datenpunkte etwas. We benutzen dazu Gewichtsfunktionen. Diese Gewichtsfunktionen werden "Kerne" genannt and with  $K$  bezeichnet. Wenn die Kerne integrierbar are, normieren we sie konventionell so, dass  $\int K(x)dx = 1$ . Einige übliche Kerne are in Tabelle 1.9 aufgelistet and in Abb. 1.1 gezeigt. Wenn sie einen kompakten Träger haben, so is als Träger das Intervall  $[-1, 1]$  gewählt (The R-Konvention is es, die Kerne so zu standardisieren, dass sie die Standardabweichung 1 haben).

Kern	$K(x)$
Uniform	$1/2$
Dreieck	$1 -  x $
Epanechnikov (quadratisch)	$3/4(1 - x^2)$
Biweight	$15/16(1 - x^2)^2$
Triweight	$35/32(1 - x^2)^3$
Gauß	$(2\pi)^{-1/2} \exp(-x^2/2)$

Table 1.9 Einige übliche Kerne

ta:ch01kernel

Durch Verschiebung and Umskalierung definiert jeder Kern eine ganze Familie

$$\frac{1}{h} K\left(\frac{x - x_0}{h}\right).$$

Bandbreite | **textbf**  
smoothing | **textbf**

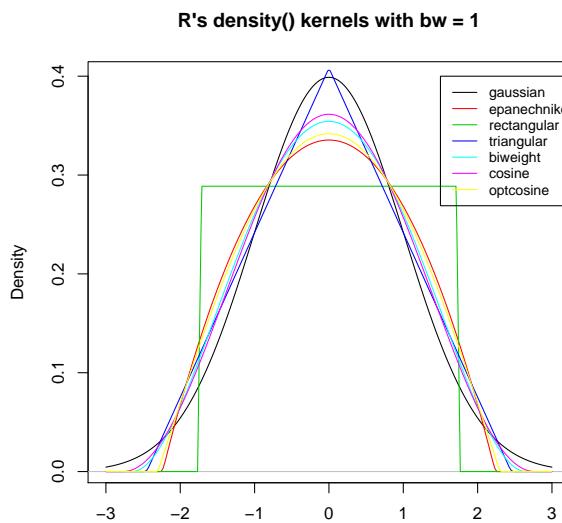


Figure 1.1 Kerne in R

**fig:Rkernels**

Der Skalenfaktor  $h$  wird **Bandbreite** genannt. Der mit  $h$  skalierte Kern wird mit  $K_h$  bezeichnet:

$$K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right).$$

The function

$$x \mapsto \frac{1}{n} \sum_i K_h(x - X_i)$$

resultiert anstelle des Histogramms ein geglättetes Bild.

Näheres dazu findet man unter dem Stichwort **smoothing** in der Literatur.

### 1.3.1 First Pass for Example S01:ex01 I.I: Distribution Functions

**subs:s01-vf1**

Um zu prüfen, ob eine Zufallsfolge zu einer distribution with distribution function  $F$  passt, vergleiche man  $F$  with  $F_n$ . Im Fall der uniformen distribution auf  $(0, 1)$  ist  $F(x) = F_{unif}(x) = x$  for  $0 \leq x \leq 1$ . Der ganz naive Zugang berechnet die Functions  $F_n$  and  $F$ . Eine erste Überlegung sagt:  $F_n$  is eine stückweise konstante function with Sprungstellen an den Beobachtungspunkten. We bekommen also ein vollständiges Bild von  $F_n$ , wenn we  $F_n$  an den Beobachtungspunkten  $X_i, i = 1..n$  ausvaluesn. Ist  $X_{(i)}$  die  $i$ . Ordnungsstatistik, so is - bis auf Bindungen -  $F_n(X_{(i)}) = i/n$ . We vergleichen  $F_n(X_{(i)})$  with dem "Sollwert"  $F(X_{(i)}) = X_{(i)}$ . Eine R-Implementierung, with Hilfsvariablen notiert:

```
n <- 100
x <- runif(n)
xsort <- sort(x)
i <- (1:n)
y <- i/n
plot(xsort, y)
```

Eine zusätzliche Gerade für die "Sollvalues" kann with

```
abline(0, 1)
```

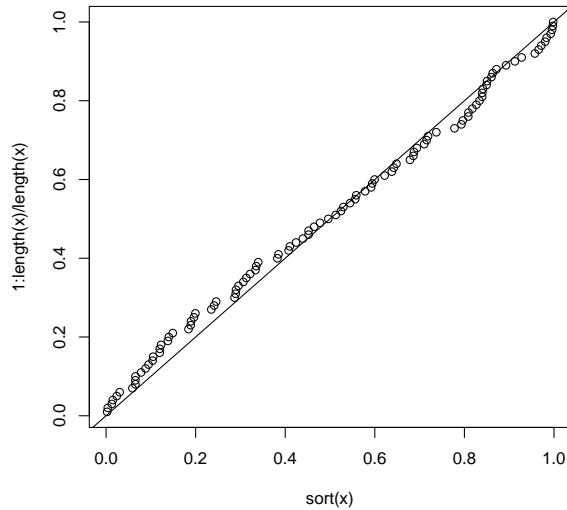
eingezzeichnet werden.

Eine kompaktere Implementierung with der function `length()`:

**Example 1.2:**

*Input*

```
x <- runif(100)
plot(sort(x), 1:length(x)/length(x))
abline(0, 1)
```



**R Functions**

(Fortsetzung)→

<code>plot@plot text plot@plot text plotmath expression@exp</code>	<b>R Functions</b> (Fortsetzung)	
	<code>sort()</code>	Sortiert vector <i>Beispiel:</i> <code>sort(runif(100))</code>
	<code>length()</code>	Länge eines Vektors <i>Beispiel:</i> <code>length(x)</code>
	<code>abline()</code>	Fügt Linie in Plot hinzu <i>Beispiel:</i> <code>abline(a = 0, b = 2)</code>

The function `plot()` fügt defaultmäßig Beschriftungen hinzu. Damit die Grafik für sich aussagekräftig ist, wollen wir diese durch genauere Beschriftungen ersetzen. Dazu ersetzen wir die Default-parameter von `plot()` durch unsere eigenen. Der parameter `main` kontrolliert die Hauptüberschrift (Default: leer). Wir können diese zum Beispiel ersetzen wie in

```
plot(sort(x), (1:length(x))/length(x),
     main = "Empirische distribution function\n (X uniform)").
```

Mit `xlab` und `ylab` wird die Beschriftung der Achsen gesteuert. Über diese und weitere Parameter kann man Information mit `help(plot)` abfragen, werden dann aber weiter an `help(title)` verwiesen.

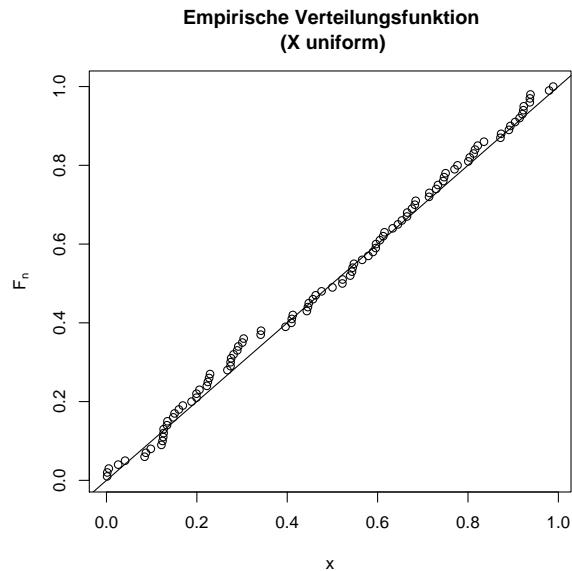
The vertikale Achse gibt noch eine Herausforderung: mit `ylab = "Fn(x)"` als Parameter würden wir eine Beschriftung mit  $Fn(x)$  erhalten. The übliche Bezeichnung setzt aber den Stichprobenumfang als Index, also  $F_n(x)$ . Hier hilft eine versteckte Eigenschaft der Beschriftungsfunktionen: Wird als Parameter eine Zeichenkette übergeben, so wird sie ohne Umwandlung angezeigt. Wird als Parameter ein R-Ausdruck übergeben, so wird versucht, die mathematisch übliche Darstellung zu geben. Details findet man mit `help(plotmath)` und Examples mit `demo(plotmath)`. The Umwandlung einer Zeichenkette in einen (unausgevaluesten) R-Ausdruck geschieht mit `expression()`.

expl:ch01-1.3

**Example 1.3:***Input*

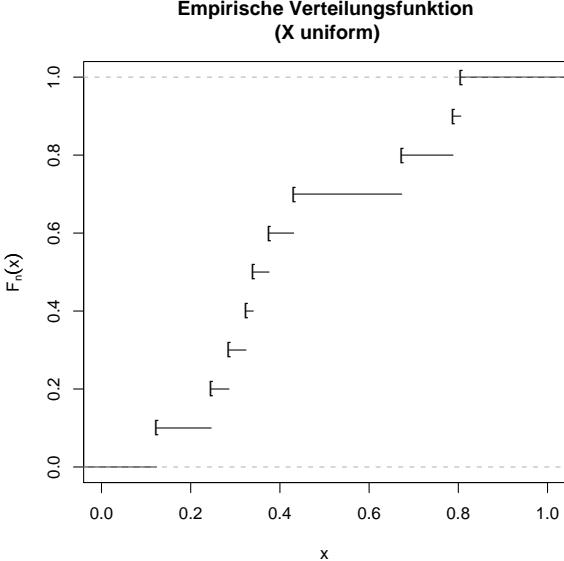
```
x <- runif(100)
plot(sort(x), (1:length(x))/length(x),
     xlab = "x", ylab = expression(F[n]),
     main = "Empirische Verteilungsfunktion\n(X uniform)")
)
abline(0, 1)
```

plot@plot|textit



Dieses Beispiel dient hier nur zur Einführung. Es ist nicht notwendig, die distribution function selbst zu programmieren. In R gibt es z.B. die Klasse `ecdf` für die empirische distribution function. Wird die function `plot()` auf ein Objekt der Klasse `ecdf` angewandt, so führt die “generische” function `plot()` intern auf die spezielle function `plot.ecdf`, und diese zeichnet in der for Verteilungsfunktionen speziellen Weise. Wir können das Beispiel also abkürzen durch den Aufruf `plot(ecdf(runif(100)))`.

a:ch01:01-ecdf

Aufgabe 1.3	Ergänzen Sie den Aufruf <code>plot(ecdf( runif(10) ))</code> durch weitere parameter so, das die Ausgabe die folgende Form hat: 
-------------	--

a:ch01:01-3

Aufgabe 1.4	Mit <code>rnorm()</code> generieren Sie gaußverteilte random variables. Versuchen Sie, anhand der Serienplots gaußverteilte random variables von uniform verteilten zu unterscheiden. Benutzen Sie dann die empirischen Verteilungsfunktionen. Können Sie damit gaußverteilte von uniform verteilten unterscheiden? The Sinus-Serie von uniform verteilten? von gaußverteilten? Wie groß ist der benötigte Stichprobenumfang, um die distributions verlässlich zu unterscheiden?
-------------	---

1.3.2 First Pass for Example S01:ex02

subs:S01hist1

We wählen Testmengen  $A_j$ ,  $j = 1, \dots, J$  im valuesbereich von  $X$ . Strategie: Um zu prüfen, ob eine Zufallsfolge zu einer distribution  $P$  gehört, vergleiche man den vector  $(P(A_j))_{j=1,\dots,J}$  with  $(P_n(A_j))_{j=1,\dots,J}$ . for die uniforme distribution auf  $(0, 1)$  können we

z.B. die Intervalle

$$A_j = \left( \frac{j-1}{J}, \frac{j}{J} \right]$$

als Testmengen wählen. Dann ist

$$(P(A_j))_{j=1,\dots,J} = (1/J, \dots, 1/J)$$

der theoretische Vergleichsvektor zum vector der beobachteten relative Häufigkeiten  
 $\frac{\#i:X_i \in A_j}{n} \quad j = 1, \dots, J$ . Vorläufige Implementierung: we benutzen hier gleich eine vorgefertigte function, die Histogramme zeichnet. Als Seiteneffekt liefert sie uns die gewünschten values. Mit der function `rug()` können we die Originaldaten zusätzlich einblenden.

expl:ch01-fg4hist

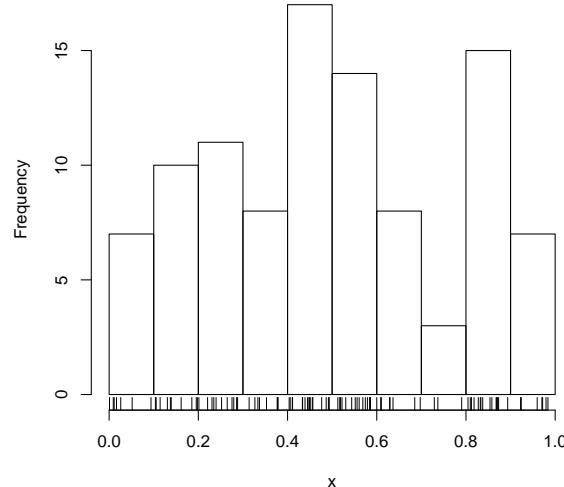
```
rug@rug|textit
density@density|textit
hist@hist|textit
```

**Example 1.4:**

*Input*

```
x <- runif(100)
hist(x)
rug(x)
```

Histogram of x



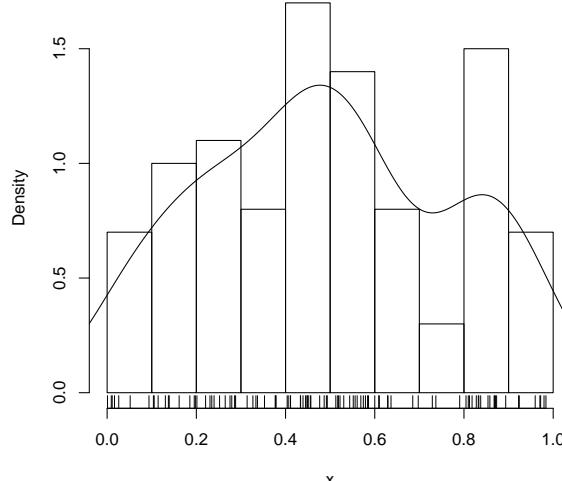
Zum Vergleich können we einen Dichteschätzer überlagern. Da `density()` im Gegensatz zu `hist()` das Resultat nicht zeichnet, sondern ausdrückt, müssen we die Grafik explizit anfordern. Damit die Skalen vergleichbar are, fordern we for das Histogramm with dem parameter `probability = TRUE` eine probabiliydsdarstellung an.

01-fg4histkernel

**Example 1.5:**

*Input*

```
hist(x, probability = TRUE)
rug(x)
lines(density(x))
```

**Histogram of x**

Histogramm und Kern-Dichteschätzer haben jeweils ihre spezifischen Vorteile und Probleme. Histogramme leiden unter ihrer Diskretisierung, die mit einer Diskretisierung in den Daten unglücklich zusammen treffen kann. Kern-Dichteschätzer "verschmieren" die Daten, und können dadurch insbesondere am Rand des Datenbereichs zu unangemessenen Rand-Effekten führen.

Zurück zum Histogramm: Benutzen wir eine Zuweisung

```
xhist <- hist(x),
```

so wird die interne Information des Histogramm unter `xhist` gespeichert und kann mit

```
xhist
```

abgerufen werden. Sie resultiert in z.B.

expl:ch01-fig5hist

**Example 1.6:**

<pre> x &lt;- runif(100) xhist &lt;- hist(x) xhist </pre>	<i>Input</i>
<pre> \$breaks [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0  \$counts [1] 12 12  3  7 11  7 14  8 16 10  \$intensities [1] 1.200000 1.200000 0.300000 0.700000 1.100000 0.700000 1.400000 [8] 0.800000 1.600000 1.000000  \$density [1] 1.200000 1.200000 0.300000 0.700000 1.100000 0.700000 1.400000 [8] 0.800000 1.600000 1.000000  \$mids [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95  \$xname [1] "x"  \$equidist [1] TRUE  attr(,"class") [1] "histogram" </pre>	<i>Output</i>

*Counts* gibt dabei die Besetzungsnumbers der Histogrammzellen, d.h. die von uns gesuchte Anzahl. The in *xhist* gespeicherte interne Information des Histogramms besteht aus fünf wesentlichen Komponenten - hier jeweils vectors. Diese Komponenten von *xhist* haben names and können with help dieser names angesprochen werden. So gibt z.B.

*xhist\$counts*

den vector der Besetzungsnumbers.

<del>hist@hist text hist@hist text table@table te</del>	R Datenstrukturen
vectors	Komponenten eines Vektors werden über ihren Index angesprochen. Alle Elemente eines Vektors haben denselben Typ. <i>Beispiele:</i> <code>x</code> <code>x[10]</code>
Listen	Listen are zusammengesetzte Datenstrukturen. The Komponenten einer Liste haben names, über die sie angesprochen werden können. Teilkomponenten einer Liste können von unterschiedlichem Typ sein. <i>Beispiele:</i> <code>xhist</code> <code>xhist\$counts</code>

Weitere zusammengesetzte Datenstrukturen are im Anhang ([S0A1Datenstrukturen](#) ([A.18](#))) beschrieben.

The Wahl der Histogrammgrenzen erfolgt automatisch. for die genaue Behandlung der Intervallgrenzen gibt es unterschiedliche conventions, deren Wahl durch parameter von `hist()` gesteuert werden kann. Um unsere Testmengen zu benutzen, müssen we die Aufrufstruktur von `hist()` erfragen.

`a:ch01:e2hist`

Aufgabe 1.5	
	<p>Generieren Sie zu <code>runif(100)</code> Histogramme with 5, 10, 20, 50 gleichgroßen Zellen and ziehen Sie wiederholt Stichproben.          Entsprechen die Bilder dem, was Sie von unabhängig uniform verteilten random variables erwarten? Versuchen Sie, ihre observations möglichst genau zu notieren.          Wiederholen Sie das Experiment with zwei Zellen (0, 0.5], (0.5, 1).  <code>hist(runif(100), breaks = c(0, 0.5, 1))</code>          Wiederholen Sie das Experiment analog with <code>rnorm(100)</code> and vergleichen Sie die Resultate von <code>runif(100)</code> and <code>rnorm(100)</code>.</p>

### Bar Charts

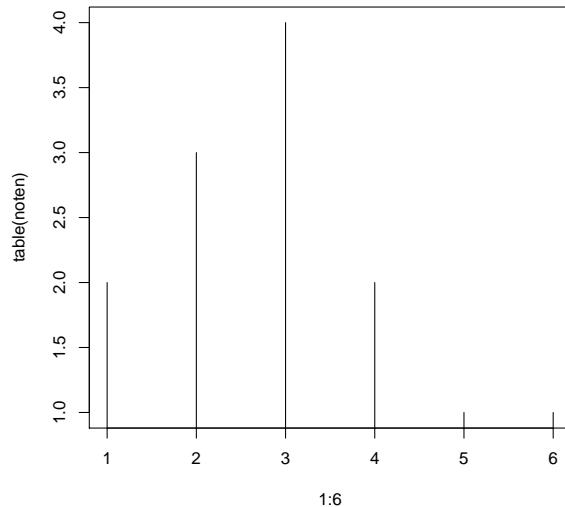
Als Hinweis: wenn die Daten nicht quantitativ are, sondern kategorial (durch Kategorien-Label bezeichnetet, wie z.B. "sehr gut, gut, befriedigend, ...", or durch Kenndigitn bezeichnetet, wie z.B. "1, 2, 3, ..."), so is ein Balkendiagramm eher geeignet. Einfache Balkendiagramme werden von `plot()` selbst durch den parameter `type = h` unterstützt. Dazu müssen aus den Rohdaten die Häufigkeiten der einzelnen Stufen bestimmt werden. Dies kann with der function `table()` geschehen.

`expl:ch01-bar`

**Example 1.7:**

*Input*

```
noten <- c(2, 1, 3, 4, 2, 2, 3, 5, 1, 3, 4, 3, 6)
plot(1:6, table(noten), type = 'h')
```



a:ch01:e3bar

Aufgabe 1.6															
	<p>Modifizieren Sie den Aufruf von <code>plot</code> im obigen Beispiel so, dass der Plot das folgende Aussehen hat:</p> <p style="text-align: center;"><b>Notenverteilung</b></p> <table border="1"> <caption>Data from Note Distribution Histogram</caption> <thead> <tr> <th>Note</th> <th>Anzahl</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2</td> </tr> <tr> <td>2</td> <td>3</td> </tr> <tr> <td>3</td> <td>4</td> </tr> <tr> <td>4</td> <td>2</td> </tr> <tr> <td>5</td> <td>1</td> </tr> <tr> <td>6</td> <td>1</td> </tr> </tbody> </table>	Note	Anzahl	1	2	2	3	3	4	4	2	5	1	6	1
Note	Anzahl														
1	2														
2	3														
3	4														
4	2														
5	1														
6	1														

**ToDo:** remove line caps - bar line is too long

### 1.3.3 Statistics of Distribution Functions; Kolmogorov-Smirnov-Tests

We machen jetzt einen Schritt von a naiven Ansatz zu einer statistischen Betrachtung. Naiv haben we for unabhängig identisch verteilte Variable  $(X_1, \dots, X_n)$  with distribution function  $F$  angenommen, dass  $i/n = F_n(X_{(i)}) \approx F(X_{(i)})$  and dies zur Überprüfung der Verteilungsannahme benutzt. Speziell for uniform  $(0, 1)$  verteilte Variable is diese naive Annahme:  $i/n \approx X_{(i)} = F(X_{(i)})$ .

Statistisch gesehen is  $X_{(i)}$  eine Zufallsvariable. Damit is auch  $F(X_{(i)})$  eine Zufallsvariable with valuesn in  $[0, 1]$ , and we können die distribution dieser random variables untersuchen.

**ToDo:**  
this is  
~~thmfs01-beta~~  
rem. make  
numerati-  
on more  
logical

**Theorem 1.4** Sind  $(X_1, \dots, X_n)$  unabhängig identisch verteilte random variables with stetiger distribution function  $F$ , so is  $F(X_{(i)})$  verteilt nach der Beta-distribution  $\beta(i, n - i + 1)$ .

*Proof.* → probabilitystheorie. Hinweis: Benutze

$$X_{(i)} \leq x_\alpha \Leftrightarrow (\#j : X_j \leq x_\alpha) \geq i.$$

for stetige distributions is  $(\#j : X_j \leq x_\alpha)$  binomialverteilt with Parametern  $(n, \alpha)$ .  $\square$

**Corollary 1.5**

$$E(F(X_{(i)})) = i/(n+1).$$

Aufgabe 1.7	
	Mit <code>help(rbeta)</code> erhalten Sie Informationen über die Functions, die for die Beta-distributions bereitstehen. Plotten Sie die entsprechenden Dichten der Beta-distributions for $n = 10, 50, 100$ and $i = n/4, n/2, 3n/4$ . Benutzen Sie zum plotten die function <code>curve()</code> . Zum Aufruf, siehe <code>help(curve)</code> .

We können also im statistischen Mittel for uniform auf  $(0, 1)$  verteilte Variable nicht erwarten, dass  $X_{(i)} \approx i/n$ , sondern im Mittel erhalten we  $i/(n+1)$ . The “richtige Sollwertgerade” sollte also with `abline(a = 0, b = n/n+1)` gezeichnet werden.

Aufgabe 1.8	
	Zeichnen Sie die distribution function with der korrigierten Geraden.
*	for die grafische Darstellung wird jeweils nur ein Plot benutzt. Ist der Erwartungswert von $X_{(i)}$ hier der richtige Vergleichsmaßstab? Gibt es Alternativen? Falls Sie Alternativen sehen: implementieren Sie diese.

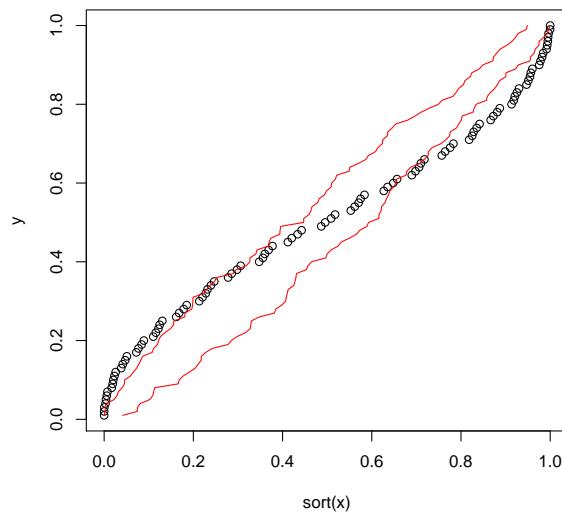
*Monte Carlo Confidence Bands*

Mit einer Simulation können we uns auch ein Bild von der typischen Fluktuation verschaffen. We benutzen random numbers, um eine (kleine) Anzahl von Stichproben bekannter distribution zu generieren, and vergleichen die in Frage stehende Stichprobe with den Simulationen. Dazu Bilden we for die Simulationen die Einhüllende, and prüfen, ob die Stichprobe innerhalb dieses Bereichs liegt. Ist  $x$  der in Frage stehende vector with Länge  $n$ , so benutzen we z.B. die folgende Programmidee:

`expl:ch01-MC06`

**Example 1.8:**

```
Input
x <- (sin(1:100)+1)/2          # demo example only
y <- (1:length(x))/length(x)
plot(sort(x), y)
nrsamples <- 19                # nr of simulations
samples <- matrix(data = runif(length(x)* nrsamples),
nrow = length(x), ncol = nrsamples)
samples <- apply(samples, 2, sort)
envelope <- t(apply(samples, 1, range))
lines(envelope[, 1], y, col = "red")
lines(envelope[, 2], y, col = "red")
```

**ToDo: apply and related**

for die Programmierung wird hier eine for R typische Strategie erkennbar. R ist eine interpretierte vektor-orientierte Sprache. Einzelne Interpretationsschritte sind zeitintensiv. Deshalb sind Operationen mit weniger, dafür komplexeren Schritten effektiver als Operationen aus mehreren elementaren Schritten.

- Operationen auf Vektorebene sind effektiver als Ketten einzelner elementare Operationen.
- Iterationen und Schleifen werden vermieden zugunsten strukturierter vector-Operationen.

a:ch01:e5

Aufgabe 1.9	Benutzen Sie die <code>help()</code> -function and kommentieren Sie das obige Beispiel Schritt for Schritt. Notieren Sie insbesondere die neu hinzugekommenen Functions.
-------------	--

est!Monte-Carlo|textbf

**ToDo:**  
 ch01: add  
 xamples  
 outer

R Iteratoren	
<code>apply()</code>	wendet eine function auf die Zeilen or Spalten einer Matrix an. <i>Beispiel:</i> <code>samples &lt;- apply(samples, 2, sort)</code> sortiert spaltenweise.
<code>outer()</code>	erzeugt eine Matrix with allen Paar-Kombinationen aus zwei vectors, and wendet eine function auf jedes Paar an.

Wenn die Kurve for unsere Stichprobe die durch die Simulation gewonnenen Grenzen überschreitet, so widerspricht das der Hypothese, dass der Stichprobe and der Simulation das selbe Modell zugrunde liegt. Das hier skizzierte Verfahren heißt **Monte-Carlo-Test**. The Idee dahinter is von sehr allgemeiner Bedeutung.

a:ch01:e6

Aufgabe 1.10	
*	Wieso 19? <i>Hinweis:</i> versuchen Sie, das Problem zunächst abstrakt and vereinfacht zu betrachten: sei $T$ eine messbare function and $X_0, X_1, \dots, X_{nrsamples}$ unabhängige Stichproben with einer gemeinsamen stetigen distribution function. Berechnen Sie $P(T(X_0) > T(X_i))$ for alle $i > 0$ . Formulieren Sie dann das obige Beispiel abstrakt. Spezialisieren Sie dann for <code>nrsamples = 19</code> .

a:ch01:e7

<code>bquote()</code>	<b>Aufgabe 1.11</b>
*	<p>Schätzen Sie die Überdeckungsprobabilität des Monte-Carlo-Bands, in dem Sie wie folgt vorgehen: Generieren Sie zunächst analog zum obigen Beispiel ein Band. (Wie können Sie das Band zeichnen, ohne zuvor für eine spezielle Stichprobe einen Plot zu machen?) Ziehen Sie für eine zu wählende Anzahl <code>sim</code> (100? 1000? 999?) jeweils eine Stichprobe von uniform verteilten random numbers vom Stichprobenumfang 100. Zählen Sie aus, wie oft die empirische distribution function der Stichprobe innerhalb des Bands verläuft. Schätzen Sie hieraus die Überdeckungsprobabilität.</p> <p>Hinweis: <code>any()</code> kann benutzt werden, um <code>for</code> einen ganzen vector einen Vergleich zu machen.</p>

**ToDo:**

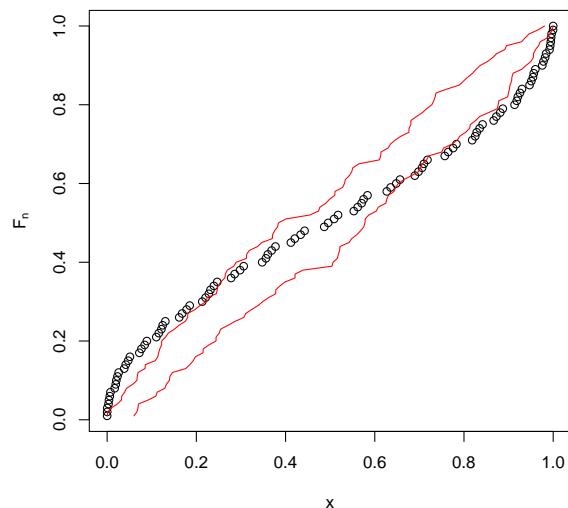
work out  
as example

We wollen auch hier die Ausgabe noch überarbeiten, so dass der Plot genügend Information enthält. Bei der Beschriftung können wir zunächst analog zu Abschnitt I.3.1 vorgehen. The Anzahl `nrsamples` bedarf des Nachdenkens. Wenn wir nur eine feste Anzahl (z.B. 19) betrachten wollen, können wir diese wie gewohnt in die Beschriftung aufnehmen. Wenn das Programmfragment jedoch allgemeiner nutzbar sein soll, so müssen wir die jeweils gewählte Anzahl von Simulationen angeben. Dies von Hand zu tun ist eine Fehlerquelle, die vermieden werden kann. The function `bquote()` ermöglicht es, den jeweils aktuellen value zu erfragen or im jeweiligen Kontext zu berechnen. Damit die Anzahl von Simulationen in die Überschrift übernommen werden kann, vertauschen wir die Anweisungen so dass die Anzahl der Simulationen vor dem Aufruf von `plot` festgelegt ist.

`expl:ch01-MC06a`

**Example 1.9:**

<pre>x &lt;- (sin(1:100)+1)/2 y &lt;- (1:length(x))/length(x) nrsamples &lt;- 19 plot(sort(x), y,      main = paste("Monte-Carlo-Band: ", bquote( .(nrsamples)), " Monte-Carlo-Samples"),      xlab = 'x', ylab = expression(F[n])) samples &lt;- matrix(data = runif(length(x) * nrsamples),                    nrow = length(x), ncol = nrsamples) samples &lt;- apply(samples, 2, sort) envelope &lt;- t(apply(samples, 1, range)) lines(envelope[, 1], y, col = "red") lines(envelope[, 2], y, col = "red")</pre>	<i>Input</i>
---	--------------

**Monte-Carlo-Band: 19 Monte-Carlo-Samples**

for die Simulationen werden jeweils neue Monte-Carlo-Stichproben gezogen. Deshalb erhalten Sie bei jedem Aufruf unterschiedliche Monte-Carlo-Konfidenzbänder und die Bänder hier are von denen im vorherigen Beispiel verschieden.

for die praktische Arbeit kann es notwendig sein, die Verteilungsdiagnostik auf ein einfaches Entscheidungsproblem zu reduzieren, etwa um anhand von Tabellen or Kontrollkarten zu entscheiden, ob eine distribution in a hypothetischen Bereich liegt, or eine Kenngröße anzugeben, die die Abweichung von a gegebenen Modell charakterisiert. Wenn we auf Tabellen or einfache numbers zurückgreifen wollen, müssen we uns weiter einschränken. We müssen die Information, die in den Functions ( $F_n, F$ ) steckt, weiter reduzieren, wenn we die Unterschiede numerisch zusammenfassen wollen. Eine Zusammenfassung is

**ToDo:**

add label for examples

etwa

$$\sup_x |F_n - F|(x).$$

Wenn we diese Zusammenfassung als Kriterium benützen wollen, stehen we wieder vor der Aufgabe, ihre distribution zu untersuchen.

**Theorem 1.6** (*Kolmogorov, Smirnov*) for stetige Verteilungsfunktionen  $F$  is die distribution von

$$\sup_x |F_n - F|(x)$$

unabhängig von  $F$  (jedoch abhängig von  $n$ ).

*Proof.* → probabilitystheorie. Z.B. [Gänßler & Stute, Lemma 3.3.8].  $\square$

**Theorem 1.7** (*Kolmogorov*): for stetige Verteilungsfunktionen  $F$  and  $n \rightarrow \infty$  hat

$$\sqrt{n} \sup |F_n - F|$$

asymptotisch die distribution function

$$F_{\text{Kolmogorov-Smirnov}}(y) = \sum_{m \in \mathbb{Z}} (-1)^m e^{-2m^2 y^2} \quad \text{for } y > 0.$$

*Proof.* → probabilitystheorie. Z.B. [Gänßler & Stute, Formel (3.3.11)].  $\square$

for die praktische Arbeit bedeutet dies: for stetige Verteilungsfunktionen können we eine Entscheidungsstrategie formulieren: we entscheiden, dass die Beobachtung  $(X_1, \dots, X_n)$  nicht with der Hypothese von unabhängig, identisch nach  $F$  verteilten random variables vereinbar is, falls  $\sup |F_n - F|$  zu groß is:

$$\sup |F_n - F| > F_{\text{krit}} / \sqrt{n},$$

wobei  $F_{\text{krit}}$  aus der (von  $F$  unabhängigen) distribution function der Kolmogorov-Smirnov-Statistik zum Stichprobenumfang  $n$  entnommen wird. Wählen we speziell das obere  $\alpha$ -Quantil  $F_{\text{krit}} = F_{\text{Kolmogorov-Smirnov}, 1-\alpha}$ , so wissen we, dass bei Zutreffen der Hypothese der value  $F_{\text{krit}}$  or ein höherer value höchstens with probabiliy  $\alpha$  erreicht wird. Damit können we unsere Irrtumsprobabiliy for eine ungerechtfertigte Ablehnung der Hypothese kontrollieren.

Asymptotisch, for große  $n$ , können we anstelle der distribution function die Kolmogorov-Approximation benutzen. Wenn die Modellverteilung  $F$  nicht stetig is, are weitere Überlegungen nötig.

We wollen uns hier auf die Programmierung konzentrieren and gehen nicht in die Details des Kolmogorov-Smirnov-Tests. Mit elementaren Mitteln können we die Teststatistik  $\sup_x |F_n - F|(x)$  for die uniforme distribution programmieren. Aus Monotoniegründen is

$$\sup_x |F_n - F|(x) = \max_{X(i)} |F_n - F| X(i)$$

and for die uniforme distribution is

$$\max_{X(i)} |F_n - F| X(i) = \max_i |i/n - X(i)|.$$

Damit gibt in R-Schreibweise der Ausdruck

```
max( abs((1: length(x)) / length(x)) - sort(x)) )
```

die für uns die gewünschte Statistik, wenn `x` unser Datenvektor ist.

Diese Statistik (und viele weitere allgemein benutzte Statistiken) are in der Regel schon programmiert, ebenso wie die zugehörigen Verteilungsfunktionen.<sup>†</sup>

a:ch01:e7ks

Aufgabe 1.12	
	<p>Mit</p> <pre>help(ks.test)</pre> <p>erhalten Sie die Information, wie die function <code>ks.test</code> angewandt wird.</p> <p>Welche Resultate erwarten Sie, wenn Sie die folgenden vectors auf uniforme distribution testen:</p> <pre>1:100 runif(100) sin(1:100) rnorm(100)?</pre> <p>Führen Sie diese Tests durch (skalieren Sie dabei die values so, dass sie im Intervall [0, 1] liegen, or benutzen Sie eine uniforme distribution auf a angepassten Intervall.) and diskutieren Sie die Resultate.</p>

#### 1.3.4 Statistics of Histograms and Related Plots; $\chi^2$ -Tests

subs:S01hist2

Wie bei der distribution function machen we einen Schritt in Richtung auf eine statistische Analyse. Der Einfachheit halber nehmen we an, dass we disjunkte Testmengen  $A_j, j = 1, \dots, J$  gewählt haben, die den valuesbereich von  $X$  überdecken. The Beobachtung  $(x_1, \dots, x_n)$  gibt dann Besetzungsnumbers  $n_j$

$$n_j = (\#i : X_i \in A_j).$$

Wenn  $(X_i)_{i=1, \dots, n}$  unabhängig are with identischer distribution  $P$ , so is  $(n_j)_{j=1, \dots, J}$  ein Zufallsvektor with Multinomialverteilung zu den Parametern  $n, (p_j)_{j=1, \dots, J}$  with  $p_j = P(A_j)$ . for den Spezialfall  $J = 2$  haben we die Binomialverteilung. Da we freie Wahl

<sup>†</sup> Unterschiedliche Implementierungen können hier andere Aufrufstrukturen vorsehen. Der Kolmogorov-Smirnov-Test findet sich in `ks.test`.

Vor R Version 2.x gehörten diese jedoch nicht zum Basis-Umfang von R, sondern are in speziellen Bibliotheken enthalten, die explizit hinzugeladen werden mussten. The Bibliothek with klassischen Tests in der R1.x-Implementierung heißt `ctest` and wird with

```
library(ctest)
```

geladen.

**Test!Median-** über die Testmengen  $A_j$  haben, können we damit eine ganze Reihe von oft hilfreichen Spezialfällen abdecken, z.B.

*Mediantest auf Symmetrie:*

$$A_1 = \{x < x_{0.5}\} \quad A_2 = \{x \geq x_{0.5}\}$$

*Midrange-Test auf Konzentration:*

$$A_1 = \{x_{0.25} \leq x < x_{0.75}\} \quad A_2 = \{x < x_{0.25} \text{ or } x \geq x_{0.75}\}.$$

für den allgemeinen Fall müssen we jedoch die empirischen Besetzungsnumbers  $n_j$  anhand der Multinomialverteilung beurteilen, and diese is sehr unangenehm zu berechnen. Deshalb greift man oft auf Approximationen zurück. Auf Pearson geht folgende Approximation zurück:

**Lemma 1.8 (Pearson):** for  $(p_j)_{j=1,\dots,J}, p_j > 0$  gilt im Limes  $n \rightarrow \infty$  die Approximation

$$\begin{aligned} P_{mult}(n_1, \dots, n_J; n, p_1, \dots, p_J; ) \approx \\ (2\pi n)^{-1/2} \left( \prod_{j=1,\dots,J} p_j \right)^{-1/2} \cdot \\ \exp \left( -1/2 \sum_{j=1,\dots,J} \frac{(n_j - np_j)^2}{np_j} \right. \\ - 1/2 \sum_{j=1,\dots,J} \frac{n_j - np_j}{np_j} \\ \left. + 1/6 \sum_{j=1,\dots,J} \frac{(n_j - np_j)^3}{(np_j)^2} + \dots \right). \end{aligned}$$

*Proof.* → probabilitystheorie. Z.B. [10] p. 285. □

**ToDo:**  
check  
pearson

Der erste Term wird bestimmt von  $\widehat{\chi^2} := \sum_{j=1,\dots,J} (n_j - np_j)^2 / np_j$ . Dieser Term wird  $\chi^2$ -Statistik genannt. Zumindest asymptotisch for  $n \rightarrow \infty$  führen große values von  $\widehat{\chi^2}$  zu kleinen probabiliyen. Dies motiviert, die  $\chi^2$ -Statistik approximativ als Anpassungsmaß zu benutzen. Der name kommt aus der Verteilungsasymptotik:

**Theorem 1.9 (Pearson):** for  $(p_j)_{j=1,\dots,J}, p_j > 0$  is im Limes  $n \rightarrow \infty$  die Statistik

$$\widehat{\chi^2} := \sum_{j=1,\dots,J} \frac{(n_j - np_j)^2}{np_j}$$

$\chi^2$ -verteilt with  $J - 1$  Freiheitsgraden.

for eine formale Entscheidungsregel können we wieder einen kritischen value  $\chi^2_{krit}$  festlegen, and die Hypothese, dass die observations  $(X_1, \dots, X_n)$  identisch uniform verteilte random numbers are, wenn die  $\chi^2$ -Statistik über diesem value liegt. Wählen we als kritischen value das obere  $\alpha$ -Quantil der  $\chi^2$ -distribution, so wissen we, dass bei Zutreffen

der Hypothese der value  $\chi^2_{krit}$  or ein höherer value höchstens with probabiliy  $\alpha$  erreicht wird. Damit können we zumindest asymptotisch auch hier unsere Irrtumsprobabiliy for eine ungerechtfertigte Ablehnung der Hypothese kontrollieren.

The  $\chi^2$ -Tests gehören zum Basisumfang von R als function `chisq.test()`. Sie are so ausgelegt, dass sie for allgemeinere "Kontingenztafeln" genutzt werden können. We benötigen sie hier nur for einen Spezialfall: die Tafel is in unserem Fall der (eindimensionale) vector der Besetzungsnumbers for vorgewählte Zellen. (Hinweis: in der R-Implementierung are allgemeinere Varianten in `library(loglin)` zu finden.)

a:ch01:e8chisq

Aufgabe 1.13	
	Orientieren Sie sich with <code>help(chisq.test)</code> über die Aufrufstruktur der $\chi^2$ -Tests. Wenden Sie ihn for die Hypothese ( $p_j = 1/J$ ), $J = 5$ auf folgende vectors von Besetzungsnumbers an:  $(3 \ 3 \ 3 \ 3 \ 3) \quad (1 \ 2 \ 5 \ 3 \ 3) \quad (0 \ 0 \ 9 \ 0 \ 6).$

a:ch01:e8achisq

Aufgabe 1.14	
	Welche Resultate erwarten Sie, wenn Sie die folgenden vectors with dem $\chi^2$ -Test auf uniforme distribution testen:  $1:100$ $runif(100)$ $sin(1:100)$ $rnorm(100)?$  Führen Sie diese Tests durch and diskutieren Sie die Resultate.  <i>Hinweis:</i> The Funktion <code>chisq.test()</code> erwartet als Eingabe eine Häufigkeitstabelle. The Prozedur <code>table()</code> gibt die Möglichkeit, Besetzungstabellen direkt zu erstellen (siehe <code>help(chisq.test)</code> ). Sie können aber auch die function <code>hist()</code> benutzen, and den Eintrag <code>counts</code> aus dem Resultat benutzen.

The Approximationen for die  $\chi^2$ -Statistik gelten zunächst nur, wenn die Zellen fest gewählt are, unabhängig von der Information aus der Stichprobe. Praktische Histogramm-Algorithmen bestimmen jedoch Zellenanzahl und Zellgrenzen aufgrund der Stichprobe. Dazu werden (implizit) parameter der distribution geschätzt. Unter bestimmten Voraussetzungen gilt noch immer eine  $\chi^2$ -Asymptotik, wie z.B nach dem folgenden Theorem aus [19]Section 6b.2:

**ToDo:**  
Rao trans-  
lation

Stichproben!wiederholte|textbf{Theorem 1.10} (i) Let the cell probabilities be the specified functions  $\pi_1(\boldsymbol{\theta}), \dots, \pi_k(\boldsymbol{\theta})$  involving  $q$  unknown parameters  $(\theta_1, \dots, \theta_q) = \boldsymbol{\theta}'$ . Further let

- (a)  $\hat{\boldsymbol{\theta}}$  be an efficient estimator of  $\boldsymbol{\theta}$  in the sense of (5c.2.6),
- (b) each  $\pi_i(\boldsymbol{\theta})$  admit continuous partial derivatives of the first order (only) with respect to  $\theta_j$ ,  $j = 1, \dots, q$  or each  $\pi_i(\boldsymbol{\theta})$  be a totally differentiable function of  $\theta_1, \dots, \theta_q$ , and
- (c) the matrix  $M = (\pi_r^{-1/2} \partial \pi_r / \partial \theta_s)$  of order  $(k \times q)$  computed at the true values of  $\boldsymbol{\theta}$  is of rank  $q$ . Then the asymptotic distribution of

$$\chi^2 = \sum \frac{(n_i - n\hat{\pi}_i)^2}{n\hat{\pi}_i} = \sum \frac{(0 - E)^2}{E} \quad (1.1) \quad \text{eq:?}$$

is  $\chi^2(k - 1 - q)$ , where  $\hat{\pi}_i = \pi_i(\hat{\boldsymbol{\theta}})$ .

Proof. Siehe rao73ls1 [19] Abschnitt 6b.2.  $\square$

a:ch01:e9

Aufgabe 1.15	
*	<p>Entwerfen Sie vergleichbare Testumgebungen für feste und für adaptive Zellwahlen.</p> <p>Ziehen Sie für feste und für adaptive Zellwahlen jeweils <math>s = 1000</math> Stichproben aus <code>runif()</code> vom Umfang 50; berechnen Sie formal die <math>\chi^2</math>-Statistik und plotten Sie deren distribution function.</p> <p>Vergleichen Sie die Verteilungsfunktionen.</p>

### ToDo:

histogram  
as density  
estimator;  
choice of  
cells  
ToDo:  
reference  
to kernel  
density  
estimators

### Wiederholte Stichproben

We haben uns bis jetzt darauf konzentriert, die distribution einer random variables zu untersuchen. We können das Verfahren fortsetzen. Wenn  $(X_1, \dots, X_n)$  identisch uniform verteilte random numbers are, dann ist bei vorgewählten Zellen die  $\chi^2$ -Statistik approximativ  $\chi^2$ -verteilt, and  $\kappa := \sqrt{n} \sup |F_n - F|$  hat asymptotisch die Kolmogorov-Smirnov-distribution.

We können wiederholt Stichproben  $(X_{1j}, \dots, X_{nj})_{j=1..m}$  ziehen and daraus Statistiken  $\widehat{\chi^2}_j$  and  $\kappa_j$  berechnen. Bei unabhängig, identisch verteilten Ausgangsdaten müssen diese nach  $\chi^2$  resp. Kolmogorov-Smirnov verteilt sein. Bei diesen wiederholten Stichproben wird nicht nur die distribution der einzelnen observations, sondern die gemeinsame distribution der jeweils  $n$  Stichprobenelemente untersucht.

Aufgabe 1.16	
	<p>Ziehen Sie für <math>n = 10, 50, 100</math> wiederholt jeweils 300 Stichproben nach <code>runif()</code>. Berechnen Sie dafür jeweils die <math>\chi^2</math>- and Kolmogorov-Smirnov-Statistik.</p> <p>(Fortsetzung)→</p>

a:ch01:e10kChiSq

Aufgabe 1.16	(Fortsetzung)
	<p>Welchen <math>\chi^2</math>-Test benutzen Sie?</p> <p>Plotten Sie die Verteilungsfunktionen dieser Statistiken and vergleichen Sie sie with den theoretischen asymptotischen distributions.</p> <p>Sprechen irgendwelche Befunde gegen die Annahme unabhängig uniform verteilter random numbers?</p> <p><i>Hinweis:</i> die Functions for den <math>\chi^2</math>- and Kolmogorov-Smirnov-Test speichern ihre Information intern als Liste. Um die names der Listenelemente zu bekommen, kann man sich ein Testobjekt generieren. Benutzen Sie z.B.</p> <pre>names(chisq.test(runif(100))).</pre>

### Güte

The uniforme distribution war in unserer Diskussion bislang die angezielte Modellverteilung, unsere “Hypothese”. We haben diskutiert, wie die unterschiedlichen Verfahren sich verhalten müssten, wenn diese Hypothese gilt. Das daraus abgeleitete Verteilungsverhalten kann dazu dienen, kritische Grenzen for formale Tests festzulegen. We verwerfen die Hypothese, wenn die beobachteten Test-Statistiken zu extrem are. Was “zu extrem” is, wird anhand der abgeleiteten distributions bestimmt. Dies führt zu Entscheidungsregeln wie:

verwerfe die Hypothese, wenn  $F_{\chi^2}(\widehat{\chi^2}) \geq 1 - \alpha$

or

verwerfe die Hypothese, wenn  $F_{Kolmogorov-Smirnov}(\kappa) \geq 1 - \alpha$   
for geeignet festzulegende (kleine) values von  $\alpha$ .

Wenn we ein Entscheidungsverfahren formal festgelegt haben, können we im nächsten Schritt fragen, wie scharf das Verfahren is, wenn die Hypothese tatsächlich abzulehnen is. Eine genauere Analyse bleibt der Statistik-Vorlesung vorbehalten. Mit den bis jetzt diskutierten Möglichkeiten können we jedoch schon das Verhalten with einer Monte-Carlo-Strategie untersuchen.

Als Simulations-Szenario wählen we eine Familie von Alternativen. The uniforme distribution fügt sich in die Beta-distributions with den Dichten

$$p_{a,b}(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1} \quad \text{for } a > 0, b > 0 \text{ and } 0 < x < 1$$

ein. We wählen als Alternativen distributions aus dieser Familie. Daraus ziehen we wiederholt Stichproben, and wenden jeweils formal unsere Entscheidungsverfahren an. We registrieren, ob das Verfahren zu einer Ablehnung der Hypothese führt or nicht. Zu gegebener Wahl eines Stichprobenumfangs  $n$  and einer Wiederholungsanzahl  $m$  and bei Wahl einer Grenzprobabilitiy  $a$  erhalten we eine Tabelle

$$(a, b) \mapsto \# \text{ Simulationen, bei denen die Hypothese verworfen wird.}$$

Speziell für die uniforme distribution  $(a, b) = (1, 1)$  erwarten wir annähernd  $m \cdot \alpha$  Verwerfungen. Für andere distributions ist ein Verfahren um so entscheidungsschärfer, je größer der Anteil der Verwerfungen ist.

a:ch01:e11ts

Aufgabe 1.17	
**	<p>Untersuchen Sie die Trennschärfe des Kolmogorov-Smirnov-Test und des <math>\chi^2</math>-Tests. Wählen Sie jeweils einen value for <math>n, m</math> and <math>\alpha</math>, and wählen Sie 9 Paare for <math>(a, b)</math>. Notieren Sie die Überlegungen hinter Ihrer Wahl.</p> <p>Ziehen Sie zu diesen Parametern with <code>rbeta()</code> Zufallsstichproben. Führen Sie jeweils den Kolmogorov-Smirnov-Test und einen <math>\chi^2</math>-Test with 10 gleichgroßen Zellen auf <math>(0, 1)</math> durch.</p>
	<p>Wählen Sie Alternativparameter <math>(a, b)</math> so, dass Sie entlang der folgenden Geraden die Entscheidungsverfahren vergleichen können:</p> <ul style="list-style-type: none"> <li>i) <math>a = b</math></li> <li>ii) <math>b = 1</math></li> <li>iii) <math>a = 1</math></li> </ul> <p>and führen Sie eine entsprechende Simulation durch.</p>
	<p>Wählen Sie Alternativparameter <math>(a, b)</math> so, dass Sie für den Bereich <math>0 &lt; a, b &lt; 5</math> die Entscheidungsverfahren vergleichen können.</p> <p>Ihre Schlüsse?</p> <p><i>Hinweis:</i> Mit <code>outer(x, y, fun)</code> wird eine function <code>fun()</code> auf alle Paare aus den valuesn von <math>x, y</math> angewandt und das Ergebnis als Resultat zurückgeliefert.</p> <p>Mit</p> $\text{contour}()$ <p>können Sie einen Contour-Plot erzeugen. Siehe <code>demo("graphic")</code>.</p>

a:ch01:e12pseudo

Aufgabe 1.18	
**	<p>Entwerfen Sie eine Prüfstrategie, um "Pseudozufallsnumbers" zu entlarven.</p> <p>(Fortsetzung)→</p>

Aufgabe 1.18	(Fortsetzung)	Erwartungswert   textbf Varianz   textbf Standardabweichung   textbf
	<p>Testen Sie diese Strategie an einfachen examplesn</p> <ul style="list-style-type: none"> <li>i) <math>x \mod m</math> for geeignete <math>m</math></li> <li>ii) <math>\sin(x)</math> <math>x = 1..100</math></li> <li>iii) ...</li> </ul> <p>Werden diese als "nicht zufällig" erkannt?</p> <p>Versuchen Sie dann, die bereitgestellten Zufallsnumbersgeneratoren zu entlarven.</p>	

## 1.4 Moments and Quantiles

Verteilungsfunktionen or Dichten are mathematisch nicht einfach zu handhaben: der Raum der Functions is im allgemeinen unendlich-dimensional and endliche geometrische Argumente or endliche Optimierungsargumente are nicht direkt anwendbar. Um die Analyse zu vereinfachen, greift man bisweilen auf endliche Beschreibungen zurück.

Historisch haben die Momente eine wichtige Rolle gespielt: probabiliyen werden als Masse-distributions interpretiert, and die Momente analog zu den Momenten der Mechanik eingeführt. Das erste Moment, entsprechend dem Schwerpunkt, heißt in der Statistik **Erwartungswert**.

**Definition 1.11** Ist  $X$  eine reellwertige Zufallsvariable with distribution  $P$ , so is der Erwartungswert von  $X$  definiert als

$$E_P(X) := E(X) := \int X dP.$$

Das zweite Moment and höhere Momente werden konventionell zentriert. for das zweite (zentrale) Moment, die **Varianz**, haben we die folgende Definition:

**Definition 1.12** Ist  $X$  eine reellwertige Zufallsvariable with distribution  $P$ , so is die Varianz von  $X$  definiert als

$$\text{Var}_P(X) := \text{Var}(X) := \int (X - E(X))^2 dP.$$

The Integralausdrücke müssen nicht immer definiert sein, d.h. die Momente müssen nicht immer existieren. Existieren sie jedoch, so geben sie eine erste Information über die distribution. Der Erwartungswert wird oft als das "statistische Mittel" interpretiert; die Wurzel aus der Varianz, die **Standardabweichung**, als "Streuung".

The Definitionen können auch auf empirische distributions angewandt werden. Dies gibt einen ersten Weg, die Momente einer unbekannten theoretischen distribution aus den Daten zu schätzen. for den Mittelwert gilt Konsistenz:

$$E_P(E_{P_n}(X)) = E_P(X),$$

**Stichprobenvarianz** textbf{t.r.} In statististischen Mittel stimmen empirischer Erwartungswert und Erwartungswert mean@mean|textit der zu Grunde liegenden distribution überein (falls definiert).

**sd@sd|textit** for die Varianz gilt diese Konsistenz nicht, sondern es gilt

$$\frac{n}{n-1} E_P ( \text{Var}_{P_n}(X) ) = \text{Var}_P(X),$$

falls  $n > 1$ . Der mathematische Hintergrund is, dass der Erwartungswert ein linearer Operator is. Er kommutiert with linearen operators. Aber die Varianz is ein quadratischer Operator, and dass macht eine Korrektur nötig, wenn man Konsistenz will. The entsprechend korrigierte Varianz wird oft als **Stichprobenvarianz** bezeichnet.

for die Schätzung der ersten beiden Momente eines vector von random numbers stehen in R Functions bereit: **mean()** schätzt den Mittelwert and **var()** die (Stichproben-)Varianz. The function **sd()** schätzt die Standardabweichung eines Vektors.

a:ch01:01-18

Aufgabe 1.19	
	<p>Generieren Sie jeweils eine Stichprobe von 100 random variables aus den distributions with den folgenden Dichten:</p> $p(x) = \begin{cases} 0 & x < 0 \\ 1 & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases}$ <p>sowie</p> $p(x) = \begin{cases} 0 & x \leq 0 \\ 2 & 0 < x \leq 1/4 \\ 0 & 1/4 < x \leq 3/4 \\ 2 & 3/4 < x \leq 1 \\ 0 & x > 1 \end{cases}$ <p>Schätzen Sie dazu Mittelwert, Varianz and Standardabweichung.</p> <p>Wiederholen Sie die Schätzung for 1000 Stichproben. Analysieren Sie die distribution von geschätztem Mittelwert, Varianz and Standardabweichung bei wiederholten Stichproben.</p>

Momente are durch einfache arithmetische Operationen zu berechnen and ihre Kombination folgt (exakt or approximierbar) einfachen Gesetzen. Sie are jedoch sehr sensitiv. The Verschiebung einer beliebig kleinen probabiliyssmasse kann sie zum Zusammenbruch bringen. for die empirische distribution bedeutet dies: stammen die beobachteten Daten zu a Anteil  $1 - \varepsilon$  aus einer Modellverteilung and zu a Anteil  $\varepsilon$  aus einer anderen distribution, so können die Momente jeden beliebigen value annehmen, for jeden beliebig kleinen value von  $\varepsilon$ . Quantile are gegenüber a Zusammenbruch robuster als Momente. So

müssen 50% der Daten “Ausreißer” sein, bis der der Median beeinflusst wird, während das erste Moment, der Erwartungswert, schon bei Veränderung nur eines Datenpunktes beliebige values annehmen kann.

Mit der Verfügbarkeit von programmierbaren Rechnern haben Quantile als beschreibende Größe an Bedeutung gewonnen. Ihre Berechnung setzt implizit eine Sortier-Operation voraus, is also komplexer als die Berechnung von Momenten. Auch die Regeln zur Kombination are nicht so einfach wie bei Momenten and setzt oft eine explizite Rechnung voraus. Aber with den verfügbaren technischen Mitteln is dies keine wesentliche Einschränkung.

R bietet eine Reihe von Functions, um with Quantilen zu arbeiten. `quantile()` is eine elementare function, um Quantile zu bestimmen. The function `summary()` gibt eine Zusammenfassung der Verteilungsinformation, die auch auf Quantilen basiert is.

`a:ch01:e19`

Aufgabe 1.20	
	<p>Generieren Sie jeweils eine Stichprobe von 100 random variables aus den distributions von Aufgabe <a href="#">I.19</a>.</p> <p>Schätzen Sie dazu Median, oberes and unteres Quartil.</p> <p>Wiederholen Sie die Schätzung for 1000 Stichproben. Analysieren Sie die distribution von geschätztem Median, oberen and unterem Quartil bei wiederholten Stichproben.</p>

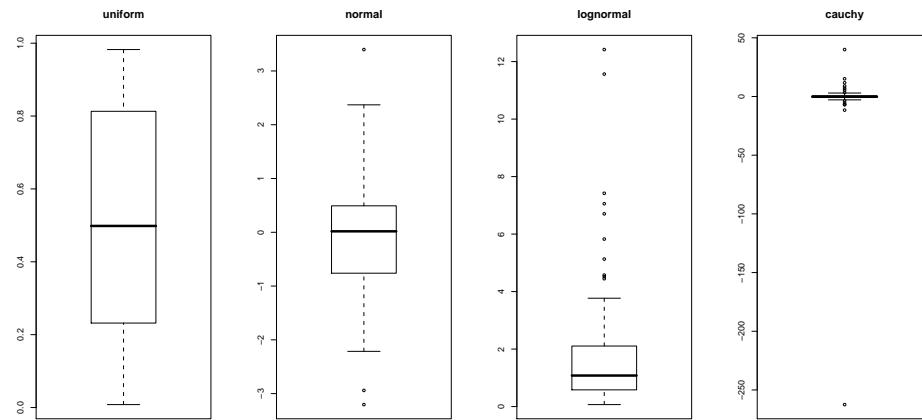
Mit `boxplot()` erhält man eine grafische Repräsentation dieser Zusammenfassung. Der hier benutzte “Box&Whisker-Plot” hat eine Reihe von Variationen. Deshalb is es bei der Interpretation notwendig, sich jeweils über die benutzten Details zu informieren. Üblich is eine Kennzeichnung durch eine “Box”, die den zentralen Teil der distribution beschreibt. In der Standardversion kennzeichnet eine Linie den Median, and eine “Box” darum reicht vom Median der oberen Hälfte bis zum Median der unteren Hälfte. Grob entspricht dies dem oberen and dem unteren Quartil. The feinere Definition sorgt dafür, dass die Information auch noch verlässlich wieder gegeben wird, wenn Bindungen, d.h. vielfache observations des selben valuess auftreten. The “Whisker” beschreiben die angrenzenden Bereiche. Ausreißer are besonders gekennzeichnet.

`expl:ch01-box`

`quantile@quantile|textit  
summary@summary|textit  
boxplot@boxplot|textit`

**Example 1.10:***Input*

```
oldpar <- par(mfrow = c(1, 4))
boxplot(runif(100), main = "uniform")
boxplot(rnorm(100), main = "normal")
boxplot(exp(rnorm(100)), main = "lognormal")
boxplot(rcauchy(100), main = "cauchy")
par(oldpar)
```



thm:s01-beta  
Theorem I.4 gibt eine Möglichkeit, Konfidenzintervalle für Quantile zu bestimmen, die allgemein gültig sind, unabhängig von der Form der zugrunde liegenden Verteilung.

Um das  $p$ -Quantil  $x_p$  einer stetigen Verteilung durch eine Rangstatistik  $X_{(k:n)}$  zum Konfidenzniveau  $1 - \alpha$  nach oben abzuschätzen, suchen wir

$$\min_k : P(X_{(k:n)} \geq x_p) \geq 1 - \alpha.$$

Aber  $X_{(k:n)} \geq x_p \iff F(X_{(k:n)}) \geq p$  und wegen Theorem I.4 ist damit

$$P(X_{(k:n)} \geq x_p) = 1 - F_{\text{beta}}(p; k, n - k + 1).$$

We können also  $\min_k$  direkt aus der Beta-Distribution ermitteln, oder wir benutzen die Beziehung zur Binomialverteilung und bestimmen  $k$  als

$$\min_k : P_{\text{bin}}(X \leq k - 1; n, p) \geq 1 - \alpha.$$

a:ch01:e4

Aufgabe 1.21	
	<p>for stetige distributions and den Verteilungsmedian <math>X_{med}</math> is <math>P(X_i \geq X_{med}) = 0.5</math>. Deshalb kann ein <math>k</math> so bestimmt werden, dass</p> $k = \min\{k : P(X_{(k)} \leq X_{med}) < \alpha\}$ <p>and <math>X_{(k)}</math> als obere Abschätzung for den Median zum Konfidenzni-veau <math>1 - \alpha</math> gewählt werden.</p> <p>Konstruieren Sie with dieser Idee ein Konfidenzintervall for den Median zum Konfidenzniveau <math>1 - \alpha = 0.9</math>.</p>
	<p>Modifizieren Sie den Box &amp; Whiskerplot so, dass er dieses Intervall einzeichnet.</p> <p><i>Hinweis:</i> Sie benötigen dazu die distribution function <math>F_X</math>, ausgeva-luest an der durch die Ordnungsstatistik <math>X_{(k)}</math> definierten Stelle. The distributions von <math>F_X(X_{(k)})</math> wird in Theorem I.4 diskutiert.</p>
	<p>Der Boxplot bietet eine Option <code>notch = TRUE</code>, um Konfidenzinter-valle zu generieren. Versuchen Sie, mithilfe der Dokumentation her-auszufinden, wie ein <code>notch</code> bestimmt wird. Vergleichen Sie Ihre Kon-fidenzintervalle with den durch <code>notch</code> gekennzeichneten Intervallen.</p>
*	<p>Bestimmen Sie analog ein verteilungsunabhängiges Konfidenzinter-vall for den Interquartilsabstand.</p>
***	<p>Ergänzen Sie den Box &amp; Whiskerplot so, dass er die Skaleninfor-mation statistisch verlässlich darstellt.</p> <p><i>Hinweis:</i> Wieso reicht es nicht, Konfidenzintervalle for die Quartile einzzeichnen?</p>

## 1.5 R Complements

s:1.3

### 1.5.1 Random Numbers

Wenn we identisch uniform verteilte random numbers hätten, könnten we auch random numbers with vielen anderen distributions generieren. Z.B.

**Lemma 1.13 (Inversionsmethode):** Ist  $(U_i)$  eine sequence unabhängiger  $U[0, 1]$  verteilt random variables and  $F$  eine distribution function, so is  $(X_i) := (F^{-1}U_i)$  eine se-quence unabhängiger random variables with distribution  $F$ .

Analytisch is dieses Lemma nur brauchbar, wenn  $F^{-1}$  bekannt is. Numerisch hilft es jedoch viel weiter: anstelle von  $F^{-1}$  werden Approximationen benutzt, oft sogar nur eine Inversionstabelle.

The Inversionsmethode is eine Methode, aus gleichverteilten random numbers andere Zielverteilungen abzuleiten. Weitere (evtl. effektivere) Methoden, aus gleichverteilten

~~PP-Plot|textbf{random numbers}~~ andere Zielverteilungen abzuleiten, werden in der Literatur zur statistischen Simulation diskutiert.

~~To Do: add refe~~ ~~QQ-Plot|textbf{for}~~ eine Reihe von distributions werden transformierte Zufallsgeneratoren bereitgestellt. Eine Liste ist im Anhang (Seite [A-93](#)) angegeben. Zu jeder Verteilungsfamilie gibt es dabei eine Reihe von Functions, deren names aus a Kurznames for die distribution abgeleitet are. for die Familie  $xyz$  is  $rxyz$  eine function, die random numbers erzeugt.  $dxyz$  berechnet die Dichte resp. das Zählmaß für diese Familie,  $pxyz$  die distribution function, and  $qxyz$  die Quantile<sup>‡</sup>.

**Übersicht:** einige ausgewählte distributions. Weitere distributions siehe [A.33](#) (Seite [A-93](#)). s:A.13

<i>distribution</i>	<i>Zufalls-numbers</i>	<i>Dichte</i>	<i>Verteilungs-funktion</i>	<i>Quantile</i>
Binomial	<i>rbinom</i>	<i>dbinom</i>	<i>pbinom</i>	<i>qbinom</i>
Hypergeometrisch	<i>rhyper</i>	<i>dhyper</i>	<i>phyper</i>	<i>qhyper</i>
Poisson	<i>rpois</i>	<i>dpois</i>	<i>ppois</i>	<i>qpois</i>
Gauß	<i>rnorm</i>	<i>dnorm</i>	<i>pnorm</i>	<i>qnorm</i>
Exponential	<i>rexp</i>	<i>dexp</i>	<i>pexp</i>	<i>qexp</i>

### 1.5.2 Graphical Comparisons

Abweichungen von einfachen geometrischen Formen werden besser wahrgenommen als Abweichungen zwischen allgemeinen Grafen ähnlicher Form. Deshalb kann es hilfreich sein, Darstellungen zu wählen, die auf einfache Formen wie z.B. Geraden führen. So wählt man um zwei Verteilungsfunktionen  $F, G$  zu vergleichen anstelle der Funktionsgraphen den Graphen von

$$x \mapsto (F(x), G(x)).$$

Dieser Graph heißt **PP-Plot** or **probability plot**. Stimmen die distributions überein, so ist der Plot eine diagonale Gerade. Abweichungen von der Diagonalgestalt are leicht zu erkennen.

Alternativ kann die Merkmalsskala als Bezug genommen werden und der Graph von

$$p \mapsto (F^{-1}(p), G^{-1}(p))$$

betrachtet werden. Dieser Graph heißt **QQ-Plot** or **Quantilplot**. Stimmen die distributions überein, so zeigt auch dieser Plot eine diagonale Gerade.

im Spezialfall der uniformen distribution auf  $[0, 1]$  ist auf diesem Intervall  $x = F(x) = F^{-1}(x)$ , d.h. **QQ-Plot** and **PP-Plot** stimmen überein and are der Graph der distribution

<sup>‡</sup> d.h. with den in der Statistik üblichen Bezeichnungen is verwirrender Weise  $p_{xyz} \equiv dxyz$  and  $F_{xyz} \equiv pxyz$ .

function. Bei nicht-uniformen distributions werden die Graphen im *PP*-Plot auf die probabiliysskala  $[0, 1]$  standardisiert, and im *QQ*-Plot auf die Merkmalsskala umskaliert.

a:ch01qqpp

Aufgabe 1.22	
	Erstellen Sie einen <i>PP</i> -Plot der $t(\nu)$ -distribution gegen die Standard-normalverteilung im Bereich $0.01 \leq p \leq 0.99$ for $\nu = 1, 2, 3, \dots$
	Erstellen Sie einen <i>QQ</i> -Plot der $t(\nu)$ -distribution gegen die Standard-normalverteilung im Bereich $-3 \leq x \leq 3$ for $\nu = 1, 2, 3, \dots$
	Wie groß muss $\nu$ jeweils sein, damit jeweils die $t$ -distribution in diesen Plots kaum von der Normalverteilung zu unterscheiden is?
	Wie groß muss $\nu$ sein, damit die $t$ -distribution bei a Vergleich der Verteilungsfunktionen kaum von der Normalverteilung zu unterscheiden is?

Können die distributions durch eine affine Transformation im Merkmalsraum ineinander überführt werden, so zeigt der Plot immer noch eine Gerade; Steigung und Achsenabschnitt repräsentieren die affine Transformation. Dies is zum Beispiel so bei der Familie der Normalverteilungen: is  $F$  die Standard-Normalverteilung  $N(0, 1)$  and  $G = N(\mu, \sigma^2)$ , so is der *QQ*-Plot eine Gerade with Achsenabschnitt  $\mu$  and Steigung  $\sigma$ .

for empirische distributions findet Korollar [I.5](#) Anwendung: anstelle von  $i/n$  wird ein for die Schiefe korrigierter Bezugspunkt gewählt, damit im Mittel eine Gerade erzeugt wird. Der Quantilplot with dieser Korrktur for empirische distributions is als function *qqplot()* bereitgestellt. for den Spezialfall der Normalverteilung is eine Variante von *qqplot()* als *qqnorm()* verfügbar, um eine empirische distribution with der theoretischen Normalverteilung zu vergleichen.

Durch die Transformationen auf dei Wahrscheinlichkeits- resp. Merkmalsskala gewinnen die graphischen Verfahren an Schärfe. So is zum Beispiel selbst bei a Stichprobenumfang von  $n = 50$  die distribution function der Normalverteilung oft nur for den geübten Betrachter von der uniformen zu unterscheiden. Im Normal-*QQ*-Plot hingegen zeigen sich uniforme Stichproben als deutlich nicht-linear, normalverteilte Daten hingegen geben weitgehend lineare Bilder.

Zur Illustration erzeugen we uns zunächst zufällige Datensätze:

*Input*

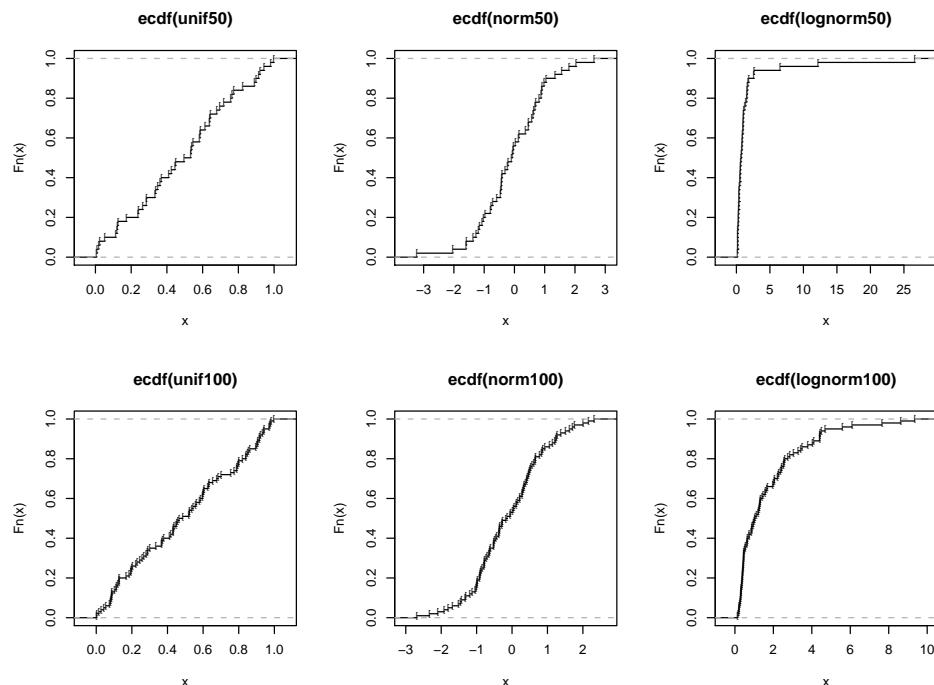
```
unif50 <- runif(50)
unif100 <- runif(100)
norm50 <- rnorm(50)
norm100 <- rnorm(100)
lognorm50 <- exp(rnorm(50))
lognorm100 <- exp( rnorm(100))
```

Mit diesen Datensätzen generieren we Plots der Verteilungsfunktionen.

expl:ch01-compecdf

**Example 1.11:***Input*

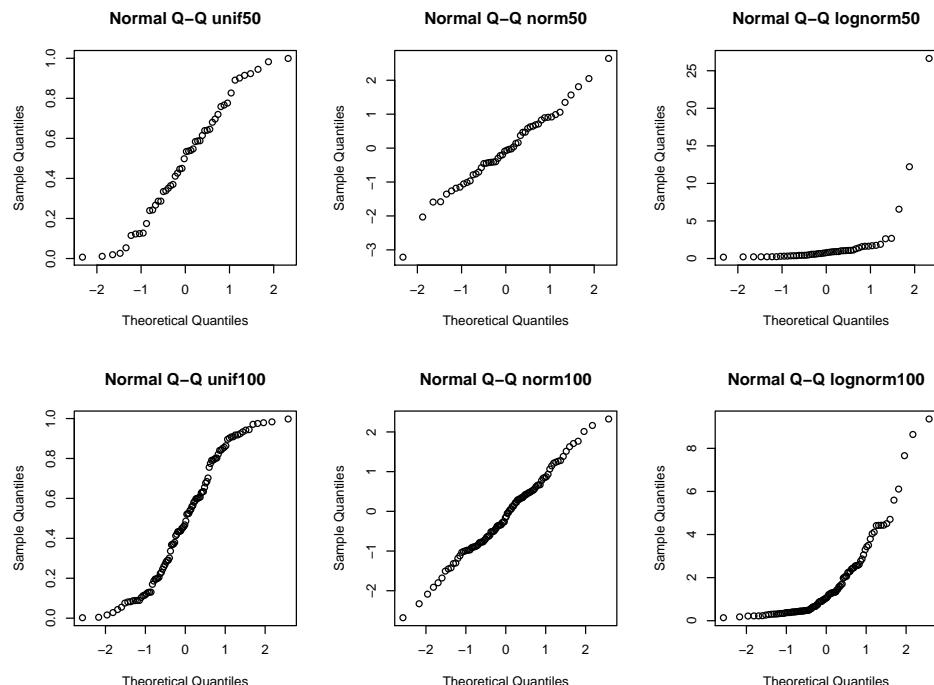
```
oldpar <- par(mfrow = c(2, 3))
plot(ecdf(unif50), pch = "[")
plot(ecdf(norm50), pch = "[")
plot(ecdf(lognorm50), pch = "[")
plot(ecdf(unif100), pch = "[")
plot(ecdf(norm100), pch = "[")
plot(ecdf(lognorm100), pch = "[")
par(oldpar)
```



**expl:ch01-compqq** Zum Vergleich dazu die entsprechenden *QQ-Plots* für die selben Daten:

**Example 1.12:***Input*

```
oldpar <- par(mfrow = c(2, 3))
qqnorm(unif50, main = "Normal Q-Q unif50")
qqnorm(norm50, main = "Normal Q-Q norm50")
qqnorm(lognorm50, main = "Normal Q-Q lognorm50")
qqnorm(unif100, main = "Normal Q-Q unif100")
qqnorm(norm100, main = "Normal Q-Q norm100")
qqnorm(lognorm100, main = "Normal Q-Q lognorm100")
par(oldpar)
```



a:ch01:e1vf

**Aufgabe 1.23**

Benutzen Sie PP-Plots anstelle von Verteilungsfunktionen, um die  $\chi^2$ - und Kolmogorov-Smirnov-Approximationen darzustellen.

a:ch01:e2qq

**ToDo:**  
better  
exercise

Aufgabe 1.24	
	Benutzen Sie <i>QQ</i> -Plots anstelle von Verteilungsfunktionen. Können Sie in diesem Plot mit help der $\chi^2$ - resp. Kolmogorov-Smirnov-Statistik Konfidenzbereiche darstellen?

Um einen Eindruck über die Fluktuation zu bekommen, müssen wir empirische Plots mit typischen Plots einer Modellverteilung vergleichen. Eine Plot-Matrix ist ein einfacher Weg dazu. Wir geben hier ein Beispiel für den Normal-*QQ*-Plot, das wir gleich als Funktion implementieren:

---

```
Input
qqnormx <- function(x, nrow = 5, ncol = 5, main = deparse(substitute(x))){
  oldpar <- par(mfrow = c(nrow, ncol))
  qqnorm(x, main = main)
  for (i in 1:(nrow*ncol-1)) qqnorm(rnorm(length(x)), main = "N(0, 1)")
  par(oldpar)
}
```

---

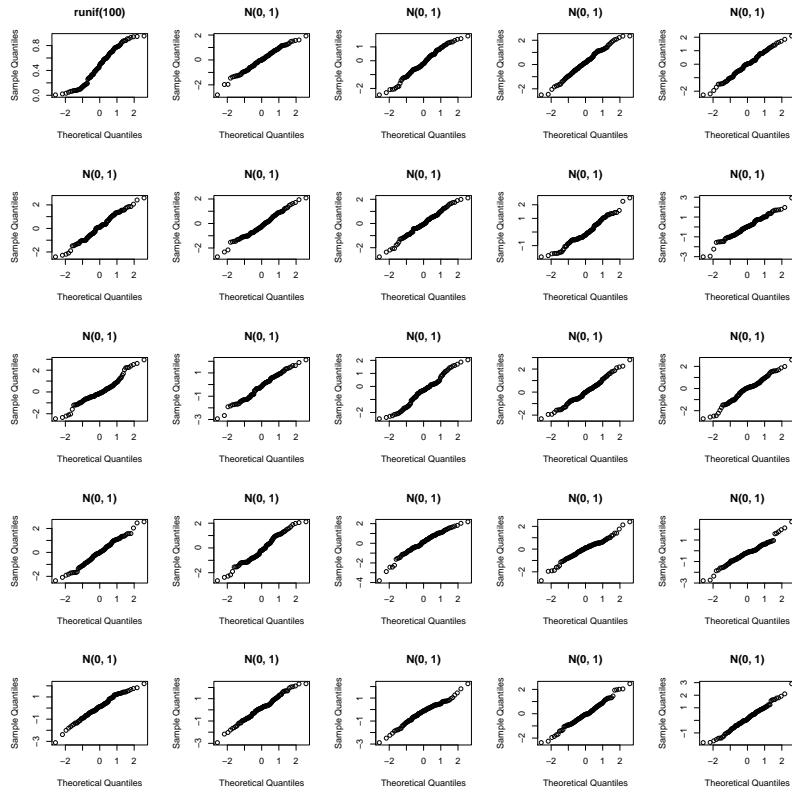
We haben in diesem Beispiel eine *for*-Schleife benutzt. Wie alle Programmiersprachen hat R Kontrollstrukturen, wie bedingte Anweisungen und Schleifen. Eine Übersicht über die Kontrollstrukturen in R ist im Anhang.

Abweichung von einer linearen Struktur ist als Fluktuation zu betrachten, wenn sie im Rahmen der simulierten examples bleibt. Ist der zu untersuchende Datensatz extrem im Vergleich zu den simulierten examples, so widerspricht das der Modellverteilung.

expl:ch01-qqmat

**Example 1.13:**  
`qqnormx(runif(100))`

Input



Auf lange Sicht lohnt es sich, die Plot-Functions so zu modifizieren, dass sie auch Informationen über die zu erwartende Fluktuation wieder geben. In Beispiel II.9 haben we for die distribution function Monte-Carlo-Bänder konstruiert. We können diese Idee auf den *PP*-Plot und den *QQ*-Plot übertragen. Dazu is es nur notwendig, die Bänder jeweils in der for den Plot geeigneten Skala darzustellen.

a:ch01:e3ppqq

Aufgabe 1.25	
	Erzeugen Sie sich with <code>rnorm()</code> Pseudozufallsnumbers for die Gauß-verteilung zum Stichprobenumfang $n = 10, 20, 50, 100$ . Erzeugen Sie jeweils einen <i>PP</i> -Plot und einen <i>QQ</i> -Plot, wobei die theoretische Gaußverteilung als Bezug dient.
	(Fortsetzung)→

legend@legend	<b>Aufgabe 1.25</b>	(Fortsetzung)
*		<p>Fügen Sie Monte-Carlo-Bänder aus der Einhüllenden von 19 Simulationen hinzu.</p> <p>Sie müssen zunächst anstelle der uniformen distribution die Normalverteilung zur Erzeugung der Monte-Carlo-Bänder benutzen. Sie müssen außerdem die Resultate im Koordinatensystem des <i>QQ</i>-Plots darstellen, d.h. die x-Achse repräsentiert die Quantile der Normalverteilung. Hinweis: inspizieren Sie dazu die Quelle von <i>qqnorm()</i>.</p> <p>The Bänder are zunächst Bänder for die Standard-Normalverteilung. Finden Sie Bänder for die vorliegenden Daten.</p>

### 1.5.3 Complements: Enhancing Graphical Displays

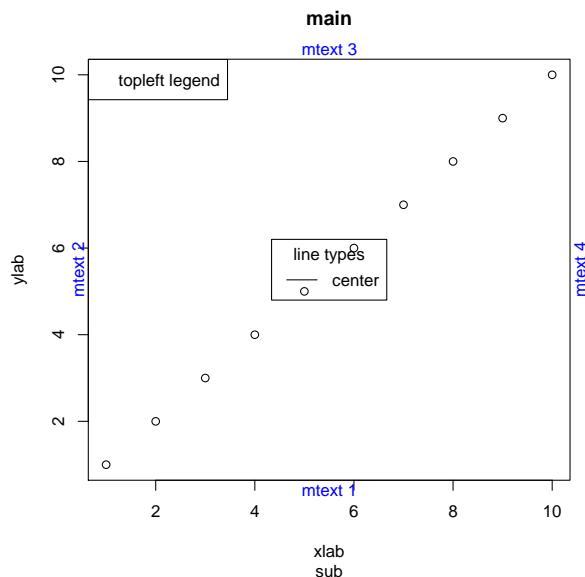
Bislang wurde die R-Grafik in rudimentärer Form benutzt. for ernsthafte Arbeit muss die Grafik so aufbereitet werden, dass ihre Bestandteile identifiziert and wiedererkennbar are. Dazu gehören Überschriften, Achsenkennzeichnungen etc. R unterscheidet zwischen “high level”-Grafik and “low level”. “High level”-Functions erzeugen eine neue Grafik. Sie bieten darüber hinaus Möglichkeiten, allgemeine Grafikparameter zu steuern.

The “low level”-Functions fügen Elemente zu vorhandenen Grafiken hinzu or modifizieren die Grafik im Detail. The function *legend()* kann Legenden innerhalb der Graphik hinzu fügen.

expl:ch01-mtext

**Example 1.14:**

```
function|textbf
plot(1:10, xlab = "xlab", ylab = "ylab", main = "main", sub = "sub")
mtext("mtext 1", side = 1, col = "blue")
mtext("mtext 2", side = 2, col = "blue")
mtext("mtext 3", side = 3, col = "blue")
mtext("mtext 4", side = 4, col = "blue")
legend("topleft", legend = "topleft legend")
legend("center", legend = "center" , lty = 1:4, title = "line types")
```



a:ch01:e4plot

Aufgabe 1.26	
	Inspizieren Sie with <code>help(plot)</code> die Steuerungsmöglichkeiten der <code>plot</code> -function. Einige Detail-Information zu den Parametern erhalten Sie erst in <code>help(plot.default)</code> . Korrigieren Sie Ihren letzten Plot so, dass er eine korrekte Überschrift trägt.

Weitere Hinweise:<sup>R:Rnotes</sup> [14] Ch. 12.

#### 1.5.4 Complements: Functions

R-Kommandos können zu Functions zusammengefasst werden. Functions können parametrisiert sein. Functions erlauben eine flexible Wiederverwendbarkeit.

**To Do:**  
Add reference to lattice

expl:ch01-envfun Beispiel for eine function

*Example 1.15:*

```
ppdemo <- function (x, samps = 19) {  
  # samps: nr of simulations  
  
  y <- (1:length(x))/length(x)  
  plot(sort(x), y, xlab = substitute(x), ylab = expression(F[n]),  
       main = "distribution function with Monte-Carlo-Band (unif.)",  
       type = "s")  
  mtext(paste(samps, "Monte-Carlo-Stichproben"), side = 3)  
  samples <- matrix(runif(length(x)* samps), nrow = length(x), ncol = samps)  
  samples <- apply(samples, 2, sort)  
  envelope <- t(apply(samples, 1, range))  
  lines(envelope[, 1], y, type = "s", col = "red");  
  lines(envelope[, 2], y, type = "s", col = "red")  
}
```

**ToDo:** copy function syntax

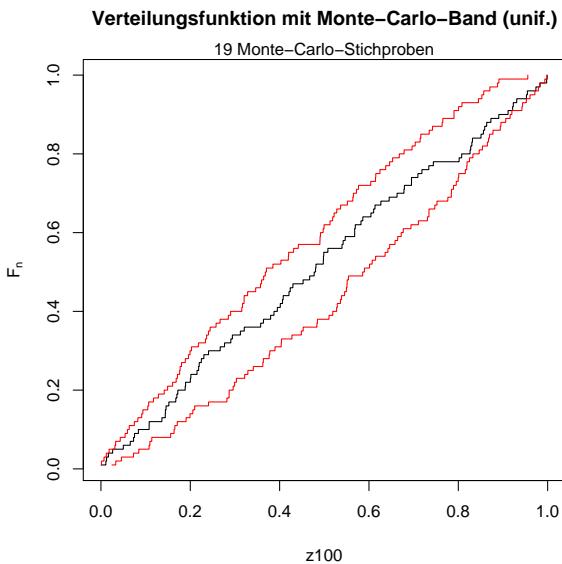
We haben bei `ppdemo()` die function `mtext()` benutzt, die Randbeschriftungen erlaubt.

Functions werden in der Form `<name>(<Aktuelle Parameterliste>)` aufgerufen.

expl:ch01-ppdemo

*Example 1.16:*

```
z100 <- runif(100)
ppdemo(z100)
```

*Input*

Wird nur der name der function eingegeben, so wird die Definition der function zurück gegeben, d.h. die function wird aufgelistet. Beispiel:

*Example 1.17:**Input*

```
ppdemo
```

*Output*

```
function (x, samps = 19) {
  # samps: nr of simulations

  y <- (1:length(x))/length(x)
  plot(sort(x), y, xlab = substitute(x), ylab = expression(F[n]),
       main = "distribution function with Monte-Carlo-Band (unif.)",
       type = "s")
  mtext(paste(samps, "Monte-Carlo-Stichproben"), side = 3)
  samples <- matrix(runif(length(x)* samps), nrow = length(x), ncol = samps)
  samples <- apply(samples, 2, sort)
  envelope <- t(apply(samples, 1, range))
  lines(envelope[, 1], y, type = "s", col = "red");
  lines(envelope[, 2], y, type = "s", col = "red")
}
```

a:ch01:e5fun

Aufgabe 1.27	
	Inspizieren Sie <code>runif()</code> with <code>qqplot()</code> and <code>plot()</code> . Überarbeiten Sie Ihre bisherigen Programmieraufgaben und schreiben Sie die wiederverwendbaren Teile als Functions.

### **ToDo:**

more  
precise  
exercise

parameter bei Functions werden dem value nach übergeben. Jede function erhält eine Kopie der aktuellen Parametervalues. Dies sorgt für eine sichere Programmierumgebung. Auf der anderen Seite führt dies zu einer Speicherbelastung and bringt einen Zeitverlust with sich. In Situationen, wo der Parameterumfang groß is or die Zeit eine kritische Größe is, kann dieser Aufwand vermieden werden, indem direkt auf Variable zugegriffen wird, die in der Umgebung der function definiert are. Entsprechende Techniken are in [R: Gentleman+Thaka:2000](#) [7] beschrieben.

Functions in R können auch geschachtelt sein, d. h. innerhalb einer function könne auch wieder Functions definiert werden. Diese are nur innerhalb der umgebenden function sichtbar.

Functions können objects als Resultate haben. Ein Objekt wird explizit als Resultat übergeben with `return(obj)`. Das Resultat kann auch implizit übergeben werden: wird das Ende einer function erreicht, ohne dass `return()` aufgerufen wurden, so wird der value des letzten ausgevaluesten Ausdruck übergeben.

<pre>circlearea &lt;- function( r ) r^2 * pi</pre>	<i>Input</i>
<pre>circlearea(1:4)</pre>	

<pre>[1] 3.141593 12.566371 28.274334 50.265482</pre>	<i>Output</i>
---	---------------

Resultate können auch bereit gestellt werde, so dass sie nur auf Anfrage übergeben werden. We haben diese Technik beim Histogramm kennen gelernt. Der Aufruf `hist(x)` übergibt kein Resultat, sondern hat nur den (gewünschten) Seiteneffekt, ein Histogramm zu zeichnen. Benutzen we `hist()` jedoch in a Ausdruck, zum Beispiel in einer Zuweisung `xhist <- hist(x)`, so erhalten we als value die Beschreibung des Histogramms. Um Resultate nur bei Bedarf zu übergeben, wird anstelle `return(obj)` der Aufruf `invisible(obj)` benutzt.

a:ch01:fun

Aufgabe 1.28		bugging parse@parse textit eval@eval textit print@print textit
	<p>Schreiben Sie als Functions:</p> <ul style="list-style-type: none"> <li>• Eine function <code>ehist</code>, die ein Histogramm with Ergänzungen zeigt.</li> <li>• Eine function <code>eecdaf</code>, die die empirische distribution zeigt.</li> <li>• Eine Funktion <code>eqqnorm</code>, die einen <i>QQ</i>-Plot with der Standard-Normalverteilung vergleicht.</li> <li>• Eine Funktion <code>eboxplot</code>, die einen Box&amp;Whisker-Plot zeigt.</li> <li>• Eine zusammenfassende function <code>eplot</code>, die eine Plot-Matrix with diesen vier Plots zeigt.</li> </ul> <p>Ihre Functions sollten die Standardfunktionen so aufrufen (or modifizieren, falls notwendig), dass die Plots eine angemessene Beschriftung erhalten.</p>	

Während Anweisungen in R schrittweise ausgeführt werden und so die Resultate bei jedem Schritt inspiziert werden können, werden beim Aufruf einer function alle Anweisungen in der function als Einheit ausgeführt. Dies kann eine Fehlersuche schwierig machen. R bietet Möglichkeiten, die Inspektion gezielt auf Functions zu ermöglichen. Details dazu finden sich im Anhang A.23 "Debugging and Profiling" auf Seite A-71.

### 1.5.5 Complements: R Internals

Ein typischer Arbeitsabschnitt von R verarbeitet Kommandos in drei Teilschritten:

- `parse()` analysiert einen Eingabetext and wandelt ihn in ein interne Darstellung als R-Ausdruck. R-Ausdrücke are spezielle R-objects.
- `eval()` interpretiert diesen Ausdruck and valuest ihn aus. Das Resultat is wieder ein R-Objekt.
- `print()` zeigt das resultierende Objekt.

Details are zu ergänzen:

#### Parse

Der erste Schritt besteht aus zwei Teilen: a Leseprozess, der die Eingabe einscannt and in Bausteine (Tokens) zerlegt, and dem eigentlichen Parsing, das die Bausteine falls möglich zu a syntaktisch korrekten Ausdruck zusammenfasst. The function `parse()` fasst beide Schritte zusammen. Dabei kann `parse()` sowohl auf lokalen Dateien arbeiten, als auch auf externen, durch eine URL-Referenz bezeichneten Dateien.

Als inverse function steht `substitute()` zur Verfügung. Eine typische Anwendung is es, aktuelle parameter eines Funktionsaufrufs zu entschlüsseln and informative Beschriftungen zu erzeugen.

#### Eval

The function `eval()` valuest einen R-Ausdruck aus. Dazu müssen die Referenzen im Ausdruck je nach den aktuellen Umgebungsbedingungen in entsprechende values übersetzt werden. Da R

`environment@environment|textit` ein interpretiertes System ist, können die Umgebungsbedingungen variieren; je nach Umgebung `search@search|textit` kann derselbe Ausdruck zu unterschiedlichen Resultaten führen.

`print@print|textit` Jede function definiert eine eigene lokale Umgebung. Functions können geschachtelt sein and `polymorph` somit auch die Umgebungen. The Umgebung kann auch dadurch verändert werden, dass Zu-`source@source|textit` pakete for R geladen werden. The aktuelle unmittelbare Umgebung kann with `environ-`  
`Sweave@Sweave|textit` erfragt werden. Mit `search()` erhält man eine Liste der Umgebungen, die sukzessive `install.packages@install.packages|textit` durchsucht werden, um Referenzen aufzulösen. Mit `ls()` erhält man eine Liste der objects in einer Umgebung.

The Erweiterbarkeit von R bringt die Möglichkeit with sich, dass Bezeichnungen kollidieren and damit die Übersetzung von Referenzen in aktuelle values fraglich wird. Als Schutz dagegen bietet R 2.x die Möglichkeit, Bezeichnungen in (geschützten) nameräumen zusammen zu fassen. In den meisten Fällen is dies transparent for den Benutzer; die Auflösung von names folgt der Suchreihenfolge, die durch die Kette der Umgebungen bestimmt is. Um explizit auf objects eines bestimmten nameraums zuzugreifen, kann dieser with angegeben werden (z. B. `base::pi` als expliziter name for die Konstante `pi` im nameraum `base`).

### *Print*

The function `print()` is als **polymorphe** function implementiert. Um `print()` auszuführen bestimmt R anhand der Klasse des zu druckenden Objekts eine geeignete Methode. Details folgen später in Abschnitt 2.6.5 Seite 2-42.

### *Executing Files*

The function `source()` steht bereit, um eine Datei als Eingabe for R zu benutzen. Dabei kann die Datei lokal sein, or über eine URL-Referenz bezeichnet sein. Konventionell wird for die names von R-Kommandodateien die Endung `.R` benutzt.

The function `Sweave()` erlaubt es, Dokumentation and Kommandos miteinander zu verweben. Konventionell wird for die names von `Sweave()`-Eingabedateien die Endung `.Rnw` benutzt. Details zum Formt finden sich in der `Sweave()`-Dokumentation

`<http://www.ci.tuwien.ac.at/~leisch/Sweave/Sweave-manual-20060104.pdf>.`

### *1.5.6 Complements: Packages*

`subs:ch01-lib`

Functions, examples, Datensätze etc. können in R zu Paketen zusammengefasst werden, die bestimmten conventions entsprechen. The conventions unterscheiden sich bei verschiedenen Implementierungen. Als aktuelle Referenz sollten die conventions von R [18] benutzt werden, denen we auch hier folgen. Eine Reihe von Paketen are Standardbestandteil von R. Pakete for spezielle Zwecke findet man im Internet z.B über <http://www.cran.r-project.org/src/contrib/PACKAGES.html>.

Nicht-Standard-Pakete müssen zunächst im R-System installiert werden. In der Regel gibt es dazu betriebssystem-spezifische Kommandos. Komfortabler is jedoch die Installation aus R with der Functions `install.packages()`. Ist keine spezielle Quelle angegeben, so greift `install.packages` dabei auf eine vorbereitete Adresse (in der Regel die oben angegebene) zurück. Sie können Pakete jedoch von jedem beliebigen Speicher laden. Insbesondere kann with

`install.packages(<package>, repos = NULL)` unter `<package>` ein direkter Zugriffspfad auf Ihrem Rechner angegeben werden.

The function `update.packages()` vergleicht installierte Versionen with dem aktuellen Stand im Netz und frischt gegebenenfalls die installierte Version auf.

Installierte Pakete werden with

`library(pkgname)`

geladen. Danach are die im Paket definierten objects (Functions, Datensätze, ...) über den aktuellen Suchpfad auffindbar and direkt verwendbar.

Pakete werden with

`detach(pkgname)`

wieder frei gegeben, d.h. ihre objects erscheinen nicht mehr im Suchpfad.

update.packages@update.package  
package.skeleton@package  
package.skeleton@package  
package.skeleton@package

Technisch are Pakete Verzeichnisse, die den R-conventions folgen. Üblich liegen sie in gepackter Form als .tar.gz-Files vor. In der Regel wird man zunächst als Benutzer vorbereitete Binärpakete installieren. Nur selten muss man auf die Quellpakete anderer Entwickler zurückgreifen.

[ToDo:](#)  
[search](#)  
[path](#)

Bei der Organisation der eigenen Arbeit lohnt es sich, den R-conventions zu folgen and zusammen gehörende Teile als R-Pakete zu organisieren. Dann stellt R eine ganze Reihe von Werkzeugen zur Unterstützung bereit. The conventions and die bereitgestellten Werkzeuge are in [18] dokumentiert. for Unix/Linux/Mac OS X-Benutzer are die wichtigsten Werkzeuge als Kommandos verfügbar:

`R CMD check <directory> # überprüft ein Verzeichnis`  
`R CMD build <directory> # generiert ein R-Paket`

Als Einstieg: The function `package.skeleton()` hilft bei der Konstruktion neuer Pakete. `package.skeleton()` erzeugt dabei außer a vorbereiteten Paket eine Hilfsdatei, die die weiteren Schritte zur Erzeugung eines ladbaren Pakets beschreibt.

Pakete müssen eine Datei DESCRIPTION with bestimmter Information enthalten. The Details are in [18] beschrieben, and ein Prototyp wird von `package.skeleton()` erzeugt. Weiteres is optional.

<i>name</i>	<i>Art</i>	<i>Inhalt</i>
DESCRIPTION	Datei	eine Herkunftsbeschreibung nach Formatkonventionen.
R	Verzeichnis	R code. Dateien in diesem Verzeichnis sollten with <code>source()</code> gelesen werden können. Empfohlene nameendung: .R .
data	Verzeichnis	Zusätzliche Daten. Dateien in diesem Verzeichnis sollten with <code>data()</code> gelesen werden können. Empfohlene nameendungen and Formate: .R for R-Code. Alternativ: .r .tab for Tabellen. Alternativ: .txt, .csv

(cont.)→

<i>name</i>	<i>Art</i>	<i>Inhalt</i>
		.RData for Ausgaben von <code>save()</code> . Alternativ: .rda. Das Verzeichnis sollte eine Datei <b>00Index</b> with einer Übersicht über die Datensätze enthalten.
exec	Verzeichnis	Zusätzliche ausführbare Dateien, z.B. Perl- or Shell-Skripte.
inst	Verzeichnis	Wird (rekursiv) in das Zielverzeichnis kopiert. Dieses Verzeichnis kann insbesondere eine Datei CITATION enthalten, die in R with einer function <code>citation()</code> ausgevaluert wird.
man	Verzeichnis	Dokumentation im R-Dokumentationsformat <code>R:Ext</code> (siehe: [18] "Writing R extensions", zugänglich über <a href="http://www.cran.r-project.org/">http://www.cran.r-project.org/</a> ). Empfohlene nameendung: .Rd
src	Verzeichnis	Fortran, C and andere Quellen.
demo	Verzeichnis	ausführbare examples. Dieses Verzeichnung sollte in einer Datei <b>00Index</b> eine Beschreibung enthalten.

a:ch01:e6lib

Aufgabe 1.29	
	<p>Installieren Sie die Functions der letzten Aufgaben als Paket. Das Paket sollte enthalten:</p> <ul style="list-style-type: none"> <li>• Eine function <code>ehist</code>, die ein Histogramm with Ergänzungen zeigt.</li> <li>• Eine function <code>eecdf</code>, die die empirische distribution zeigt.</li> <li>• Eine Funktion <code>eqqnorm</code>, die einen QQ-Plot with der Standard-Normalverteilung vergleicht</li> <li>• Eine Funktion <code>eboxplot</code>, die einen Box&amp;Whisker-Plot zeigt.</li> <li>• Eine zusammenfassende function <code>eplot</code>, die eine Plot-Matrix with diesen vier Plots zeigt.</li> </ul> <p>Sie können das Paket with <code>package.skeleton()</code> vorbereiten, wenn Sie die einzelne Functions definiert haben.</p> <p>Laden Sie dieses Paket. Überprüfen Sie, ob Sie das Paket auch nach Neustart wieder mit <code>library()</code> laden können.</p> <p><i>Hinweis:</i> is <i>x</i> ein Objekt, so erzeugt die function <code>prompt(x)</code> ein Gerüst, aus dem eine Dokumentation for <i>x</i> entwickelt werden kann.</p>

## 1.6 Statistical Summary

Als Leitbeispiel diente in diesem Kapitel die statistische Analyse einer (univariaten) Stichprobe. Dabei haben wir eine in der Statistik zentrale Modellvorstellung benutzt: die Werte der Stichprobe werden als Zufallsvariable aufgefasst, die aus einer zugrundeliegenden theoretischen Distribution entstammen. Ziel der statistischen Analyse ist der Schluss aus der empirischen Distribution der Stichprobe auf die unbekannte zu Grunde liegende theoretische Distribution. Dieser Schluss kann zwei Formen annehmen: wir können die empirische Distribution mit einer hypothetischen Distribution vergleichen. Dies ist das Vorgehen der klassischen Statistik. Wir können versuchen, aus der empirischen Distribution Merkmale der zu Grunde liegenden Distribution zu extrahieren. Dies ist das Vorgehen der Datenanalyse.

Beide Wege sind eng miteinander verwandt. Das wesentliche Werkzeug für beide war hier die Untersuchung der empirischen Distribution function.

## 1.7 Literature and Additional References

R:Ext

[18] R Development Core Team (2000-2005): Writing R extensions.

Siehe: <<http://www.r-project.org/manuals.html>>.

gnst77wth

[6] Gänßler, P.; Stute, W.: Probabilitätstheorie. Heidelberg: Springer 1977.

R:Gentleman+Thaka:2000

[7] Gentleman, R.; Ihaka, R.: Lexical Scope and Statistical Computing. Journal of Computational and Graphical Statistics 9 (2000) 491–508.

Generated by Sweave from:

```
Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S01base.Rnw.tex,v  
Revision: 1.1  
Date: 2008/02/14 18:48:37  
name:  
Author: j40  
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S01base.Rnw.tex,v $  
$Revision: 1.1 $  
$Date: 2008/02/14 18:48:37 $  
$name: $  
$Author: j40 $
```



---

## CHAPTER 2

---

# Regression

---

ch:02

### 2.1 General Regression Model

**To Do:**  
add more  
regression  
graphics

Aus der Tradition experimenteller Wissenschaften stammt das Paradigma des (kontrollierten) Versuchs. Unter Versuchsbedingungen  $x$  wird ein Resultat  $y$  gemessen, zusammengesetzt aus a systematischen Effekt  $m(x)$  and a Messfehler  $\varepsilon$ .

$$y = m(x) + \varepsilon.$$

Dies ist eine ganz spezielle Betrachtungsweise; es wird nicht unvoreingenommen das gemeinsame Verhalten von  $x$  and  $y$  untersucht, sondern eine Unsymmetrie hineingesteckt:  $x$  is die “Ursache”,  $y$  (or eine Veränderung von  $y$ ) der Effekt. The “Ursache”  $x$  is in dieser Vorstellung vorgegeben or direkt kontrollierbar;  $y$  is mittelbar (über die Versuchsbedingungen) beeinflusst. In dieser Vorstellung is  $\varepsilon$  ein Messfehler, der durch geeignete Rahmenbedingungen des Versuchs möglichst klein gehalten wird and im Mittel verschwinden sollte: es sollte keinen systematischen Fehler geben.

Vom statistischen Standpunkt is der wesentliche Unterschied der Rollen von  $x$  and  $y$ , dass for  $y$  das stochastiche Verhalten mithilfe von  $\varepsilon$  modelliert wird, während  $x$  als “gegeben” angenommen wird and dafür keine Stochastik im Modell vorgesehen is.

Um einen formal überschaubaren Rahmen zu bekommen, betrachten we den Fall, dass  $x$  als vector von reellen Variablen repräsentiert werden kann,  $x \in \mathbb{R}^p$ , and dass die Messvalues ein-dimensionale reelle values are,  $y \in \mathbb{R}$ . In a stochastiche Modell kann die oben skizzierte Idee formal gefasst werden. Eine mögliche Formalisierung is es, den Messfehler  $\varepsilon$  als Zufallsvariable zu modellieren. Nehmen we ferner an, dass der Erwartungswert von  $\varepsilon$  existiert, so können we die Annahme, dass der Messfehler im Mittel verschwindet, formalisieren als  $E(\varepsilon) = 0$ .

Um den systematischen Effekt  $m$  zu untersuchen, betrachten we Messreihen. Der Index  $i, i = 1, \dots, n$ , kennzeichnet den Messpunkt, and das Modell is damit

$$\begin{aligned} y_i &= m(x_i) + \varepsilon_i \quad i = 1, \dots, n \\ \text{with} \quad x_i &\in \mathbb{R}^p \\ E(\varepsilon_i) &= 0. \end{aligned}$$

Das statistische Problem is:

schätze die function  $m$  aus den Messvaluesn  $y_i$  bei Messbedingung  $x_i$ .

Zu diesem Problem der Kurvenschätzung or “Regression” gibt es eine umfangreiche Literatur in der Statistik. We wollen uns hier auf das “computing” konzentrieren. Dazu betrachten we zunächst eine sehr vereinfachte Version des Regressionsproblems, die lineare Regression. We sentliche Aspekte lassen sich bereits an diesem Problem illustrieren.

**Respons** | textbf{Einer einheitlichen Sprechweise zuliebe nennen we  $y_i$  die **Respons** and die Komponenten von Regressor | textbf{ $x_j$  with  $j = 1, \dots, p$ } die **Regressoren**. The function  $m$  heißt die **Modellfunktion**.

lineares  
Mo-  
dell | seeModell,  
lineares | textbf{2.2 Linear Model}

Model | !lineares

lineare Re-

gressi- We begin with dem Regressionsmodell - jetzt in Vektorschreibweise\* -  
on | seeRegression,

lineare | textbf{Regression!lineare

Design-  
Matrix | textbf

$$Y = m(X) + \varepsilon$$

$Y$  with valuesn in  $\mathbb{R}^n$

$$X \in \mathbb{R}^{n \times p}$$

$$E(\varepsilon) = 0$$

(2.1) eq:regrmod

and setzen zusätzlich voraus, dass  $m$  linear is. Dann gibt es (mindestens) einen vector  $\beta \in \mathbb{R}^p$ , so dass

$$m(X) = X\beta$$

and das Regressionsproblem is jetzt reduziert auf die Aufgabe,  $\beta$  aus der Information  $(Y, X)$  zu schätzen.

Das so modifizierte Regressionsmodell

$$Y = X\beta + \varepsilon$$

$Y$  with valuesn in  $\mathbb{R}^n$

$$X \in \mathbb{R}^{n \times p}$$

$$\beta \in \mathbb{R}^p$$

$$E(\varepsilon) = 0$$

(2.2) eq:linmod

heißt **lineares Modell** or auch **lineare Regression**. The Matrix  $X$ , in der die values der Regressoren zusammengefasst is, also die Information über die Versuchsbedingungen, heißt **Design-Matrix** des Modells.

expl:02slr **Example 2.1** (Einfache lineare Regression) Wird die Versuchsbedingung durch den value einer reellen Variablen beschrieben, von der der Versuchsausgang über eine lineare Modellfunktion

$$m(x) = a + b \cdot x,$$

abhängt, so können we eine Versuchsserie with Versuchsausgang  $y_i$  bei Versuchsbedingung  $x_i$  koordinatenweise schreiben als

$$y_i = a + b \cdot x_i + \varepsilon_i. \quad (2.3) \quad \text{span style="border: 1px solid black; padding: 2px;">eq:02slrcoord$$

In Matrixschreibweise können we die Versuchsserie zusammenfassen als lineares Modell

$$Y = \underbrace{\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}}_X \cdot \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{\beta} + \varepsilon. \quad (2.4) \quad \text{span style="border: 1px solid black; padding: 2px;">eq:02slrmat$$

\* We wechseln conventions and Schreibweisen, wenn es hilfreich is. The Verwirrung gehört zu den conventions: in einigen conventions kennzeichnen Großletters Zufallsvariable, in anderen Functions, in wieder anderen vectors. The Auflösung bleibt jeweils dem Leser überlassen.

The einfache lineare Regression is ein Beispiel for lineare Modelle. The Einweg-Klassifikation is das andere Basis-Beispiel.

expl:oneway

**Example 2.2** (Einweg-Klassifikation) Zum Vergleich von  $k$  Behandlungen (insbesondere for den Spezialfall  $k = 2$ ) benutzen we Indikatorvariablen, die in einer Matrix zusammengefasst werden. The Indikatorvariable for Behandlung  $i$  steht in Spalte  $i$ . In der Regel haben we wiederholte observations  $j = 1, \dots, n_i$  unter Behandlung  $i$ , insgesamt also  $n = \sum_{i=1}^k n_i$  observations. Dem Modell

$$Y = \underbrace{\begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}}_X \cdot \underbrace{\begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_k \end{pmatrix}}_\beta + \varepsilon. \quad (2.5) \quad \text{eq:02oneway0mat}$$

entspricht in Koordinaten

$$y_{ij} = \mu_i + \varepsilon_{ij}. \quad (2.6) \quad \text{eq:02oneway0coord}$$

Dies is das typische Modell, um die Hypothese "kein Unterschied"  $\mu_1 = \dots = \mu_k$  gegen die Alternative zu testen, dass sich die Behandlungen im Mittel unterscheiden.

Der selbe Zusammenhang kann auch dargestellt werden, wenn we die Messvalues als Summe eines Grundvalues  $\mu_0$  and dazu eines Behandlungseffekts  $\mu'_i = \mu_i - \mu_0$  interpretieren. Dies entspricht in Koordinaten

$$y_{ij} = \mu_0 + \mu'_i + \varepsilon_{ij}. \quad (2.7) \quad \text{eq:02oneway1coord}$$

In Matrixschreibweise is dies

$$Y = \underbrace{\begin{pmatrix} 1 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & \dots & 0 \\ 1 & 0 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & \dots & 1 \end{pmatrix}}_{X'} \cdot \underbrace{\begin{pmatrix} \mu_0 \\ \mu'_1 \\ \mu'_2 \\ \vdots \\ \mu'_k \end{pmatrix}}_{\beta'} + \varepsilon. \quad (2.8) \quad \text{eq:02oneway1mat}$$

**Wilkinson-Rogers-Notation** | expl:oneway  
eq:02oneway0mat  
textbf{(2.5)} Beispiel 2.2 illustriert, dass die Darstellung eines Problems als lineares Modell nicht eindeutig ist. (2.5) and (2.8) are gleichwertige Darstellungen und nur aus der Anwendung kann entschieden werden, welche den Vorrang hat.

for die mathematische Analyse is die Design-Matrix  $X$  ein wesentliches Hilfsmittel. for die Datenanalyse können we R zu help nehmen, um diese Matrix (implizit) for uns zu erstellen. R versteht eine spezielle Notation, die **Wilkinson-Rogers-Notation**, with der Modelle beschrieben werden können. In dieser Notation schreiben we

$$y \sim x.$$

Der Fehlerterm wird in diesem Modell nicht notiert.

Der konstante Term wird implizit angenommen. for die Einweg-Klassifikation erhalten we also das Modell (2.8). Wenn we keinen konstanten Term wollen (also bei der Regression die Regressionsgerade durch den Ursprung geht, resp. bei der Einweg-Klassifikation das Modell (2.5) benutzt and kein Gesamtmittel vorgegeben sein soll), so haben we in Koordinatenschreibweise

$$y_i = b \cdot x_i + \varepsilon_i.$$

In der Wilkinson-Rogers-Notation muss der konstante Term explizit auf Null gesetzt werden:

$$y \sim 0 + x.$$

Weitere Regressoren können with dem Operator  $+$  gekennzeichnet werden. So entspricht  $y \sim u + v$  in Koordinaten dem Modell

$$y_i = a + b \cdot u_i + c \cdot v_i + \varepsilon_i.$$

We kommen in den Abschnitten 2.2.4 and 2.3 noch auf diese Notation zurück.

Es gibt eine umfangreiche Literatur über lineare Modelle. Das Buch “The Theory of Linear Models” von Bent Jørgensen [Jor93t1m] is besonders zu empfehlen. Es deckt den mathematischen Hintergrund dieses Kapitels weitgehend ab and enthält zahlreiche illustrierende examples.

### 2.2.1 Factors

**subs:factor-1**

Mit help der Notation zur Design- and Modellbeschreibung kann die Übersetzung zwischen einer fallorientierten Beschreibung eines Designs in eine Design-Matrix for ein lineares Modell in kanonischer Form automatisch geschehen. Bisweilen braucht die Übersetzung etwas Nachhilfe. Betrachten Sie z.B. einen Datensatz

```
y <- c( 1.1, 1.2, 2.4, 2.3, 1.8, 1.9)
x <- c( 1, 1, 2, 2, 3, 3).
```

Der vector  $x$  kann als quantitativer vector for das Regressions-Modell

$$y_i = a + b x_i + \varepsilon_i$$

als Regressor gemeint sein, or es kann in der Einweg-Klassifikation, dem Modell der Einweg-Varianzanalyse,

$$y_{ix} = \mu + \alpha_x + \varepsilon_{ix}$$

die Kennzeichnung einer Behandlungsgruppe sein. Um beide Möglichkeiten zu unterscheiden, können vectors in R als **Faktoren** definiert werden. vectors, die keine Faktoren are, werden

als quantitative Variable behandelt wie im ersten Beispiel. Faktoren werden als Kennzeichner behandelt and in der Design-Matrix in entsprechende Indikatorvariable übersetzt. So resutls in

$$y \sim x$$

das Regressionsmodell, jedoch

$$y \sim \text{factor}(x)$$

das Varianz-Modell for das Einweg-Layout.

Durch einen parameter `ordered = TRUE` kann beim Aufruf der function `factor()` die erzeugte Variable als geordnet gekennzeichnet werden. The erzeugte Variable wird dann bei den Auswertungen als ordinal skaliert behandelt.

$$y \sim \text{factor}(x, \text{ ordered} = \text{TRUE})$$

Ohne diese Kennzeichnung werden Faktoren als kategorial skaliert betrachtet.

The numbersvalues von Faktoren brauchen keine aufsteigende sequence zu sein. Sie werden (auch for ordinale Faktoren) als bloße names benutzt and durch eine laufende Nummer ersetzt. So resutls in

$$\text{factor}( c(2, 2, 5, 5, 4, 4) )$$

einen vector with drei Faktorvaluesn 1, 2, 3, die die names "2", "5" and "4" haben. Faktoren können auch durch names bezeichnet werden, z.B.

$$y \sim \text{factor} ( c("Beh1", "Beh1", "Beh2", "Beh2", "Beh3", "Beh3") )$$

The unterschiedlichen values eines Faktors nennt man *Stufen* des Faktors. Sie können with `levels()` erfragt werden, z.B.

$$\text{levels}(\text{factor}( c(2, 2, 5, 5, 4, 4) ))$$

$$\text{levels}(\text{factor}( c("Beh1", "Beh1", "Beh2", "Beh2", "Beh3", "Beh3") ))$$

## 2.2.2 Least Squares Estimation

Eine erste Idee zur Schätzung im linearen Regressionsmodell kann so gewonnen werden: Bei gegebenem  $X$  is  $E(Y) = X\beta$ , also  $X^\top E(Y) = X^\top X\beta$  and damit  $(X^\top X)^{-1} X^\top E(Y) = \beta$ . Dabei bedeutet  $X^\top$  die transponierte Matrix zu  $X$  and  $(X^\top X)^{-1}$  die (generalisierte) Inverse von  $(X^\top X)$ . The Gleichung motiviert das folgende Schätzverfahren:

$$\hat{\beta} = (X^\top X)^{-1} X^\top Y. \quad (2.9)$$

`eq:02-kqs`

Setzt man aus [2.2](#) die Modellbeziehung  $Y = X\beta + \varepsilon$  ein and benutzt, dass  $E(\varepsilon) = 0$ , so erhält man

$$E(\hat{\beta}) = E \left( (X^\top X)^{-1} X^\top (X\beta + \varepsilon) \right) = \beta, \quad (2.10)$$

`eq:02-unbiased`

d.h.  $\hat{\beta}$  is ein erwartungstreuer Schätzer for  $\beta$ . Ob and wieweit dieser Schätzer neben dieser Konsistenz auch noch statistische Qualitäten hat, wird in Statistik-Vorlesungen diskutiert. Ein Satz zur Charakterisierung dieses Schätzers is dort als Gauß-Markov-Theorem bekannt. We werden auf diesen Schätzer häufig zurückkommen and geben ihm deshalb einen names: **Gauß-Markov-Schätzer**. Im Fall eines linearen Modells, wie dem Regressionsmodell, hat dieser Schätzer eine Reihe von Optimalitätseigenschaften. So minimiert dieser Schätzer die mittlere quadratische Abweichung, is also in diesem Modell ein **Kleinste-Quadrat-Schätzer**.

Der Kleinste-Quadrat-Schätzer for lineare Modelle wird durch die function `lm()` berechnet.

Zu Illustration erzeugen we uns einen Beispiel-Datensatz.

```
factor@factor|textit
Skala!ordinal
Skala!kategorial
Stufen|seeFaktor,
      Stufen|textbf
Faktor!Stufen
Gauß-
  Markov-
    Schätzer|textbf
Kleinste-
  Quadrat-
    Schätzer|textbf
lm@lm|textit
```

**Fit|textbf**


---

```
Input  
x <- 1:100  
err <- rnorm(100, mean = 0, sd = 10)  
y <- 2.5*x + err
```

**bsp:02-lm**

Den Kleinste-Quadrate-Schätzer erhalten we nun durch

**Example 2.1:**

---

*Input*  
`lm(y ~ x)`


---

*Output*  
**Call:**  
`lm(formula = y ~ x)`
  
**Coefficients:**  

(Intercept)	x
-2.642	2.536

**a:ch02:lm01**

<b>Aufgabe 2.1</b>	
	We haben die Daten ohne konstanten Term erzeugt, dies aber bei der Schätzung nicht vorausgesetzt. Wiederholen Sie die Schätzung im Modell ohne konstanten Term. Vergleichen Sie die Resultate.

Der Schätzer  $\hat{\beta}$  führt unmittelbar zu einer Schätzung  $\hat{m}$  for die function  $m$  in unserem ursprünglichen Modell:

$$\hat{m}(x) = x^\top \cdot \hat{\beta}.$$

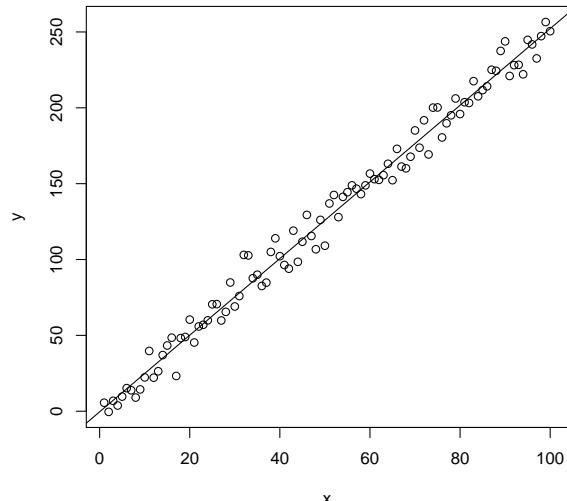
The Auswertung an a Punkt  $x$  resultls in values  $\hat{y} := \hat{m}(x)$ , den **Fit** an der Stelle  $x$ . The Auswertung an den Messpunkten resultls in den vector der gefitteten values  $\hat{Y} = X\hat{\beta}$ .

In unserem Beispiel is dies eine Regressionsgerade. Mit `plot()` können we die Datenpunkte zeichnen. Wenn we das Resultat der Regression speichern, können we with `abline()` die Regressionsgerade hinzufügen.

**expl:02-lmplot**

**Example 2.2:**

```
lmres <- lm(y ~ x)
plot(x, y)
abline(lmres)
```

*Input*Hut-  
Matrix|textbf

`abline()` is a function, die Geraden anhand von unterschiedlichen Parametrisierungen zeichnen kann. Weiter Information erhalten sie with `help(abline)`.

The Schätzgleichung (2.9) gibt uns an, wie der Fit an den Messpunkten berechnet wird.

$$\hat{Y} = X(X^\top X)^{-1}X^\top \cdot Y. \quad (2.11)$$

eq:02-fit

The Matrix

$$H := X(X^\top X)^{-1}X^\top \quad (2.12)$$

eq:02-hat

nennt man **Hut-Matrix**<sup>†</sup>. Sie ist das wesentliche Werkzeug, um den Gauß-Markov-Schätzer für eine bestimmte Design-Matrix  $X$  zu untersuchen. The Design-Matrix, and damit die Hutmatrix, hängt nur von den Versuchsbedingungen ab, nicht aber von dem Ausgang des Versuchs. Der Fit hingegen bezieht sich auf eine bestimmte Stichprobe, die in den beobachteten Stichprobewerten  $Y$  repräsentiert ist.

Im linearen Modell ist ein Term  $\varepsilon$  enthalten, der den Messfehler or die Versuchsvariabilität repräsentiert. Diesen stochastischen Fehler können we nicht direkt beobachten - sonst könnten we ihn subtrahieren and damit die Modellfunktion exakt bestimmen. We können nur mittelbar darauf schließen.

The values der Zufallsbeobachtung  $Y$  unterscheidet sich in der Regel vom Fit  $\hat{Y}$ . The Differenz

$$R_X(Y) := Y - \hat{Y}$$

**To Do:**  
mention  
unbiased

<sup>†</sup> sie setzt dem  $Y$  den Hut auf.

**Residuum** heißt **Residuum**. Das Residuum kann als Schätzer für den nicht-beobachtbaren Fehlerterm  $\varepsilon$  angesehen werden. The Residuen are nicht wirklich der Fehlerterm. Dies wäre nur der Fall, wenn die Schätzung exakt wäre. for den allgemeinen Fall zeigt uns die Beziehung

$$\begin{aligned} R_X(Y) &= Y - \hat{Y} \\ &= (I - H)Y \\ &= (I - H)(X\beta + \varepsilon) \\ &= (I - H)\varepsilon, \end{aligned} \tag{2.13} \quad \boxed{\text{eq:02-ewerr}}$$

dass die Residuen Linearkombinationen der Fehler are. We müssen aus diesen Linearkombinationen auf die Fehler zurück schließen.

Existiert die Varianz der Fehlerterme, so is durch die Varianzmatrix  $\Sigma$  der Fehlerterme  $Var(\varepsilon) = \Sigma$  die Varianz der Residuen bestimmt:

$$\begin{aligned} Var(R_X(Y)) &= Var((I - H)\varepsilon) \\ &= (I - H)\Sigma(I - H)^\top. \end{aligned} \tag{2.14} \quad \boxed{\text{eq:02-varerr}}$$

Bislang haben we nur vorausgesetzt, dass kein systematischer Fehler vorliegt, modelliert als die Annahme

$$E(\varepsilon) = 0.$$

We sprechen von a **einfachen linearen Modell**, wenn darüber hinaus gilt:

$$\begin{aligned} (\varepsilon_i)_{i=1,\dots,n} &\text{ are unabhängig} \\ Var(\varepsilon_i) = \sigma^2 &\text{ for ein } \sigma \text{ das nicht von } i \text{ abhängt.} \end{aligned}$$

Im linearen Modell versuchen we, den Parametervektor  $\beta$  zu schätzen. The Varianzstruktur des Fehlervektors bringt dabei Störparameter with sich, die die Schätzung verkomplizieren können. Im einfachen linearen Modell reduziert sich die Situation auf nur einen unbekannten Störparameter  $\sigma$ . Formeln wie eq:02-varerr vereinfachen sich, denn in diesem Fall is  $\Sigma = \sigma^2 I$  and der parameter  $\sigma$  kann aus der Formel herausgezogen werden. We könnnen diesen parameter aus den Residuen schätzen, denn die **residuelle Varianz**

$$s^2 := \frac{1}{n - Rk(X)} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \tag{2.15}$$

is ein erwartungstreuer Schätzer for  $\sigma^2$ . We schreiben deshalb auch  $\widehat{\sigma^2} := s^2$ . (Das Wurzelziehen is keine lineare Operation and erhält deshalb nicht den Erwartungswert. The residuelle Standardabweichung  $\sqrt{s^2}$  is kein erwartungstreuer Schätzer for  $\sigma$ .) Wieder in die Schätzformel (2.9) eingeschätzt liefert uns dies auch eine Schätzung for die Varianz/Covarianzmatrix des Schätzers for  $\beta$ , denn im einfachen Modell is

$$Var(\hat{\beta}) = \sigma^2 X^\top X \tag{2.16}$$

and kann durch  $s^2 X^\top X$  geschätzt werden.

The Standard-Ausgabe in Beispiel 2.1 listet nur minimale Information über den Schätzer. Mehr Information über Schätzer, Residuen and daraus abgeleitete Kenngrößen erhalten we, wenn we eine zusammengefasste Darstellung anfordern.

*Example 2.3:*

<pre>summary(lm( y ~ x))</pre>	<i>Input</i>
<pre>Call: lm(formula = y ~ x)</pre>	<i>Output</i>
<b>Residuals:</b>	
<pre>    Min      1Q  Median      3Q     Max  -25.8076 -5.4772 -0.4311  5.0360 28.6568</pre>	
<b>Coefficients:</b>	
<pre>            Estimate Std. Error t value Pr(&gt; t )     (Intercept) -2.64199   2.02938  -1.302   0.196     x             2.53573   0.03489  72.681  &lt;2e-16 ***   --- Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1</pre>	
Residual standard error: 10.07 on 98 degrees of freedom	
Multiple R-squared: 0.9818, Adjusted R-squared: 0.9816	
F-statistic: 5283 on 1 and 98 DF, p-value: < 2.2e-16	

a:ch02:1

Aufgabe 2.2	
	<p>In Abschnitt 2.3 werden wir den theoretischen Hintergrund bereitstellen, der uns hilft, die noch offenen Terme zu interpretieren.</p> <p>Der Aufruf der Funktion <code>lm()</code> liefert immer ein Resultat - wenn es den Daten angemessen ist, aber es gibt auch ein Resultat, wenn das lineare Modell gar nicht angemessen ist. Wir brauchen deshalb eine Diagnostik, die uns hilft, zu erkennen, ob das Modell verlässlich und brauchbar ist.</p> <p><b>a:ch02:1a</b></p>

**s:2.2**  
Analysieren Sie die in example 2.3 (page 2-8) gezeigten Ausgaben von `lm()`. Welche Terme können Sie interpretieren? Stellen Sie diese Interpretationen schriftlich zusammen. Für welche Terme fehlt Ihnen noch Information? Erstellen Sie eine kommentierte Version der Ausgabe.

In Abschnitt 2.3 werden wir den theoretischen Hintergrund bereitstellen, der uns hilft, die noch offenen Terme zu interpretieren.

Der Aufruf der Funktion `lm()` liefert immer ein Resultat - wenn es den Daten angemessen ist, aber es gibt auch ein Resultat, wenn das lineare Modell gar nicht angemessen ist. Wir brauchen deshalb eine Diagnostik, die uns hilft, zu erkennen, ob das Modell verlässlich und brauchbar ist.

Aufgabe 2.3	
	<p>Sei <math>yy &lt; -2.5 * x + 0.01x^2 + err</math>. Welches Resultat erhalten Sie, wenn Sie eine Regression mit dem (falschen) Modell <math>yy \sim x</math> rechnen? Gibt es Hinweise darauf, dass dieses Modell nicht angemessen ist?</p>

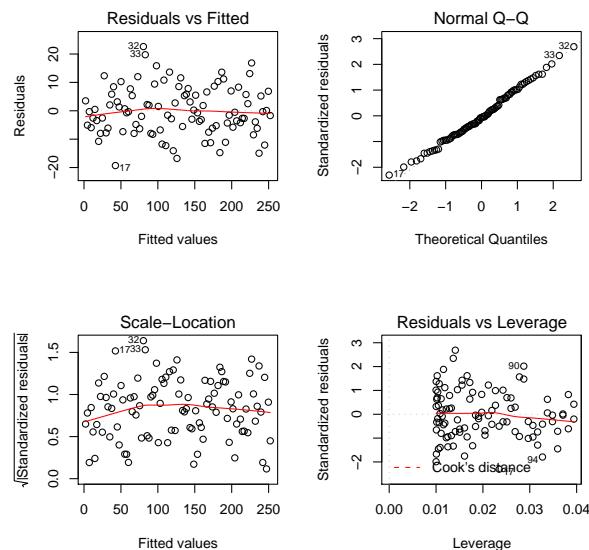
The function `lm()` führt nicht nur die Schätzung im linearen Modell durch, sondern liefert eine

ganze Reihe von Diagnostiken, die helfen können zu beurteilen, ob die Modellvoraussetzungen vertretbar erscheinen. Eine Darstellung with `plot()` zeigt vier Aspekte davon.

`expl:02-plotlm`

*Example 2.4:*

*Input*  
`plot(lm(y ~ x))`



Der obere linke Plot zeigt die Residuen gegen den Fit. Er gibt eine erste Übersicht. In der eindimensionalen Situation würde ein Plot der Residuen gegen den Regressor ausreichen. for  $p$  Regressoren wird die graphische Darstellung problematisch. Der Plot der Residuen gegen den Fit verallgemeinert sich auch auf höhere Dimensionen. The distribution der gefitteten values hängt vom Design ab.

The Residuen sollten annähernd wie ein Scatterplot von unabhängigen Variablen aussehen. The distribution der Residuen sollte nicht vom Fit abhängen. Sind hier systematische Strukturen zu erkennen, so ist das ein Warnzeichen dass das Modell or die Modellvoraussetzungen nicht erfüllt are.

Nach der vorausgegangen Diskussion können we noch genauer sein: die Residuen sollten nach (2.13) Linearkombinationen von unabhängig identisch verteilten Variablen sein. Falls die Modellvoraussetzungen erfüllt are, is die Varianz durch (2.14) beschrieben.

In der eindimensionalen Situation würde ein Plot der Residuen gegen den Regressor ausreichen. for  $p$  Regressoren wird die graphische Darstellung problematisch. Der Plot der Residuen gegen den Fit verallgemeinert sich auch auf höhere Dimensionen.

Verteilungsaussagen über die Schätzer können wir machen, wenn wir mit Verteilungsaussagen über die Fehlerterme beginnen. Die kräftigsten Aussagen sind möglich, wenn die Fehlerterme unabhängig identisch normalverteilt sind. Der obere rechte Plot sollte annähernd wie der "normal probability plot" von normalverteilten Variablen aussehen, wobei das "annähernd" wiederum bedeutet: bis auf Transformation mit der Matrix  $I - H$ .

**a:ch02:1b** The beiden übrigen Plots sind spezielle Diagnostiken für lineare Modelle (siehe `help(plot.lm)`).

```
lm@lm{textbf
print.lm@print.lm
(lm)
Topic
re-
gres-
sion!lm@lm
aov@aov{textit
as.data.frame@as.data.frame
```

Aufgabe 2.4	
	Inspizieren Sie das Resultat von Aufgabe 2.3 grafisch. Welche Hinweise gibt es jetzt, dass das lineare Modell nicht angemessen ist? <b>a:ch02:1a</b>

`plot()` stellt für lineare Modelle noch weitere diagnostische Plots bereit. Diese müssen explizit mit dem Parameter `which` angefordert werden.

---

### help(lm)

---

**lm** *Fitting Linear Models*

---

#### Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

#### Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

#### Arguments

- |                |   |
|----------------|---|
| <b>formula</b> | a symbolic description of the model to be fit. The details of model specification are given below.  |
| <b>data</b>    | an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called. |
| <b>subset</b>  | an optional vector specifying a subset of observations to be used in the fitting process.   |
| <b>weights</b> | an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with weights <code>weights</code> (that is, minimizing <code>sum(w*e^2)</code> ); otherwise ordinary least squares is used.                    |

```

options@options{textit
na.fail@na.fail{textit
na.omit@na.omit{textit
na.exclude@na.exclude{textit
model.matrix.default@model.matrix.default{possible
offset@offset{textit
model.offset@model.offset{textit
offset@offset{textit
model.matrix@model.matrix{textit
aov@aov{textit
formula@formula{textit
model, x, y, qr
lm.fit@lm.fit{textit
class@class{textit

singular.ok
contrasts
offset
...

```

a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The “factory-fresh” default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful. the method to be used; for fitting, currently only `method = "qr"` is supported; `method = "model.frame"` returns the model frame (the same as with `model = TRUE`, see below).  
logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.  
logical. If `FALSE` (the default in S but not in R) a singular fit is an error.  
an optional list. See the `contrasts.arg` of `model.matrix.default`.  
this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length either one or equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if both are specified their sum is used. See `model.offset`.  
additional arguments to be passed to the low level regression fitting functions (see below).

### Details

Models for `lm` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the cross of `first` and `second`. This is the same as `first + second + first:second`.

If the formula includes an `offset`, this is evaluated and subtracted from the response.

If `response` is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See `model.matrix` for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms` object as the formula (see `aov` and `demo(glm.vr)` for an example).

A formula has an implied intercept term. To remove this use either `y ~ x - 1` or `y ~ 0 + x`. See `formula` for more details of allowed formulae.

`lm` calls the lower level functions `lm.fit`, etc, see below, for the actual numerical computations. For programming only, you may consider doing likewise.

All of `weights`, `subset` and `offset` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

### Value

`lm` returns an object of class "lm" or for multiple responses of class `c("mlm", "lm")`.

The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

An object of class "lm" is a list containing at least the following components:

<code>coefficients</code>	a named vector of coefficients
<code>residuals</code>	the residuals, that is response minus fitted values.
<code>fitted.values</code>	the fitted mean values.
<code>rank</code>	the numeric rank of the fitted linear model.
<code>weights</code>	(only for weighted fits) the specified weights.
<code>df.residual</code>	the residual degrees of freedom.
<code>call</code>	the matched call.
<code>terms</code>	the <code>terms</code> object used.
<code>contrasts</code>	(only where relevant) the contrasts used.
<code>xlevels</code>	(only where relevant) a record of the levels of the factors used in fitting.
<code>offset</code>	the offset used (missing if none were used).
<code>y</code>	if requested, the response used.
<code>x</code>	if requested, the model matrix used.
<code>model</code>	if requested (the default), the model frame used.

In addition, non-null fits will have components `assign`, `effects` and (unless not requested) `qr` relating to the linear fit, for use by extractor functions such as `summary` and `effects`.

### Using time series

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting NAs would invalidate the time series attributes, and if NAs are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or differenced regressor would be ignored. It is good practice to prepare a `data` argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and fitted values are time series.

### Note

Offsets specified by `offset` will not be included in predictions by `predict.lm`, whereas those specified by an offset term in the formula will be.

### Author(s)

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson & Rogers (1973).

```
anova@anova|textit
terms@terms|textit
effects@effects|textit
ts.intersect@ts.intersect
predict.lm@predict.lm|tex
```

```

summary.lm@summary.lm|textit
anova.lm@anova.lm|textit
aov@aov|textit
coef@coef|textit Chambers, J. M. (1992) Linear models. Chapter 4 of Statistical Models in S eds J. M.
effects@effects|textit Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.
residuals@residuals|textit Wilkinson, G. N. and Rogers, C. E. (1973) Symbolic descriptions of factorial models for
fitted@fitted|textit analysis of variance. Applied Statistics, 22, 392–9.
vcov@vcov|textit
predict.lm@predict.lm|textit
predict@predict|textit
confint@confint|textit
lm.influence@lm.influence|textit
glm@glm|textit
lm.fit@lm.fit|textit summary.lm for summaries and anova.lm for the ANOVA table; aov for a different interface.
lm.wfit@lm.wfit|textit generic functions coef, effects, residuals, fitted, vcov.

predict.lm (via predict) for prediction, including confidence and prediction intervals;
confint for confidence intervals of parameters.
lm.influence for regression diagnostics, and glm for generalized linear models.
The underlying low level functions, lm.fit for plain, and lm.wfit for weighted
regression fitting.

```

### Examples

```

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
anova(lm.D9 <- lm(weight ~ group))
summary(lm.D90 <- lm(weight ~ group - 1))# omitting intercept
summary(resid(lm.D9) - resid(lm.D90)) #= residuals almost identical

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

## model frame :
stopifnot(identical(lm(weight ~ group, method = "model.frame"),
model.frame(lm.D9)))

```

---

Nicht erwähnt in der Help-Information: with *first-second* werden Terme der ersten Gruppe ins Modell aufgenommen, die der zweiten Gruppe aber ausgeschlossen. Ausführlichere Information zur Formel-Darstellung erhält man with *help(formula)*. Eine Zusammenfassung ist im Anhang A.53 (Seite A-85) zu finden.

The Hut-Matrix is eine Besonderheit linearer Modelle. Fit and Residuum jedoch are allgemeine Konzepte, die bei allen Arten der Schätzung angewandt werden können. The Anwender are oft with dem Fit (or der Schätzung) zufrieden. for den ernsthaften Anwender and for den Statistiker are die Residuen oft wichtiger: sie weisen darauf hin, was vom Modell or der Schätzung noch nicht erfasst is.

## 2.2.3 More Examples for Linear Models

Design-  
Matrix|textbf

subs:02-BspLinMod

The Matrix  $X$  heißt die **Design-Matrix** des Modells. Sie kann die Matrix sein, die with den ursprünglichen Messbedingungen  $x_i$  als Zeilenvektoren gebildet wird. Aber sie is nicht auf diesen Spezialfall beschränkt. Unter der scheinbar so einfachen Modellklasse der linearen Modelle lassen sich viele wichtige Spezialfälle einordnen. Ein paar davon are im folgenden zusammengestellt.

*Einfache lineare Regression:*

$$y_i = a + b x_i + \varepsilon_i \quad \text{with } x_i \in \mathbb{R}, a, b \in \mathbb{R}$$

kann als lineares Modell with

$$X = (1 \ x)$$

geschrieben werden, wobei  $1 = (1, \dots, 1)^\top \in \mathbb{R}^n$ .

*Polynomiale Regression:*

$$y_i = a + b_1 x_i + b_2 x_i^2 + \dots + b_k x_i^k + \varepsilon_i \quad \text{with } x_i \in \mathbb{R}, a, b_j \in \mathbb{R}$$

kann als lineares Modell with

$$X = (1 \ x \ x^2 \ \dots \ x^k)$$

geschrieben werden, wobei  $x^j = (x_1^j \dots x_n^j)^\top$ .

Analog for eine Vielzahl von Modellen, die durch andere Transformationen erreicht werden können.

*Varianzanalyse: Einweg-Layout*

Gemessen wird unter  $m$  Versuchsbedingungen, dabei  $n_j$  Messungen unter Versuchsbedingung  $j, j = 1, \dots, m$ . The Messung setze sich additiv zusammen aus a Grundeffekt  $\mu$ , a for die Bedingung  $j$  spezifischen Beitrag  $\alpha_j$ , and a Messfehler nach

$$y_{ij} = \mu + \alpha_j + \varepsilon_{ij} \quad \text{with } \mu, \alpha_j \in \mathbb{R}, j = 1, \dots, n_i.$$

Mit  $n = \sum n_j$  and

$$X = (1 \ I_1 \ \dots \ I_m),$$

wobei  $I_j$  die vectors der Indikatorvariablen for die Zugehörigkeit zur Versuchsgruppe  $j$  are, lässt sich dies als lineares Modell schreiben.<sup>‡</sup>

*Covarianzanalyse*

Analog zur Varianzanalyse werden Unterschiede zwischen Gruppen untersucht, aber zusätzliche (linear eingehende) Einflussfaktoren werden korrigierend berücksichtigt. Unter Versuchsbedingung  $j$  bei Beobachtung  $i$  hängt die Messung zusätzlich von Einflussfaktoren  $x_{ij}$  der Versuchseinheit  $ij$  ab.

$$y_{ij} = \mu + \alpha_j + b x_{ij} + \varepsilon_{ij} \quad \text{with } \mu, \alpha_j \in \mathbb{R}.$$

<sup>‡</sup> Es is Konvention, dass bei Varianzanalysen der letzte Index die Beobachtung zählt, and Indizes in alphabetischer sequence vergeben werden. Konventionell werden also im Vergleich zu unserer Notation die Rollen von  $i$  and  $j$  vertauscht.

**2.2.4 Model Formulae**

R erlaubt es, Modelle auch dadurch zu spezifizieren, dass die Regeln angegeben werden, nach denen die Design-Matrix gebildet wird. The Syntax, nach denen die Regeln notiert werden, is sehr kurz in der Beschreibung von `lm()` angegeben. We diskutieren sie jetzt etwas ausführlicher. Diese Modell-Spezifikation is auch for allgemeinere, nicht lineare Modelle möglich. The Modell-Spezifikationen werden als Attribut with dem names “formula” gespeichert. Sie können with `formula()` manipuliert werden.

*examples*

$$y \sim 1 + x$$

entspricht  $y_i = (1 \ x_i)(\beta_1 \ \beta_2)^\top + \varepsilon$

$$y \sim x$$

Kurzschreibweise for  $y \sim 1 + x$  (ein konstanter Term wird implizit angenommen)

$$y \sim 0 + x$$

entspricht  $y_i = x_i \beta + \varepsilon$

$$\log(y) \sim x_1 + x_2$$

entspricht  $\log(y_i) = (1 \ x_{i1} \ x_{i2})(\beta_1 \ \beta_2 \ \beta_3)^\top + \varepsilon$  (ein konstanter Term wird implizit angenommen)

$$lm(y \sim poly(x, 4), data = Experiment)$$

analysiert den Datensatz “Experiment” with a linearen Modell for polynomiale Regression vom Grade 4 in  $x$ .

Wichtige Spezialfälle for faktorielle Designs are:

$$y \sim A$$

Einweg-Varianzanalyse with Faktor  $A$ ,

$$y \sim A + x$$

Covarianzanalyse with Faktor  $A$  and Regressions-Covariable  $x$ ,

$$y \sim A + B$$

Zwei-Faktor-Kreuz-Layout with Faktoren  $A$  and  $B$  ohne Interaktion,

$$y \sim A * B$$

Zwei-Faktor-Kreuz-Layout with Faktoren  $A$  and  $B$  and allen Interaktionen (Kombinationen der Stufen von  $A$  and  $B$ ),

$$y \sim A/B$$

Zwei-Faktor hierarchisches Layout with Faktor  $A$  and Subfaktor  $B$ .

Eine Übersicht über alle operators zur Modellspezifikation is im Anhang [A.53](#) (Seite [A-85](#)) zu finden.

Aufgabe 2.5	
	Schreiben Sie die vier oben in Abschnitt <a href="#">2.2.3</a> genannten Modelle als R-Modellformeln. <small>subs:02-BspLinMod</small>
	(Fortsetzung)→

Aufgabe 2.5	(Fortsetzung)
	Erzeugen Sie sich für jedes dieser Modelle ein Beispiel durch Simulation und wenden Sie <code>lm()</code> auf diese examples an. Vergleichen Sie die durch <code>lm()</code> geschätzten parameter with den Parametern, die Sie in der Simulation benutzt haben.

The Modellformel wird in a Eintrag im Resultat von `lm()` gespeichert. Sie kann also aus dem Resultat zurück gewonnen werden. Anhand der Formel-Notation generiert R implizit eine Design-Matrix. Mit `model.matrix()` kann diese Design-Matrix inspiert werden.

a:ch02:3

Aufgabe 2.6	
	<p>Generieren Sie drei vectors with je 10 <math>N(\mu_j, 1)</math>-verteilten random variables <math>\mu_j = j, j = 1, 3, 9</math>. Verketten Sie diese zu a vector <math>y</math>.</p> <p>Generieren Sie sich einen vector <math>x</math> aus je 10 wiederholten valuesn <math>j, j = 1, 3, 9</math>.</p> <p>Berechnen Sie die Gauß-Markov-Schätzer in den linearen Modellen <math>y \sim x</math> and <math>y \sim factor(x)</math>.</p> <p>Lassen Sie sich das Resultat jeweils als Tabelle with <code>summary()</code> and als Grafik with <code>plot()</code> anzeigen and vergleichen Sie die Resultate.</p>

### 2.2.5 Gauss-Markov Estimator and Residuals

We werfen nun einen genaueren Blick auf den Gauß-Markov-Schätzer. Kenntnisse aus der linearen Algebra, langes Nachdenken or andere Quellen sagen uns:

rem:2.1

#### Remark 2.3

- (1) The Design-Matrix  $X$  definiert eine Abbildung  $\mathbb{R}^p \rightarrow \mathbb{R}^n$  with  $\beta \mapsto X\beta$ .  
Der Bild-Raum dieser Abbildung sei  $\mathcal{M}_X$ ,  $\mathcal{M}_X \subset \mathbb{R}^n$ .  $\mathcal{M}_X$  is der von den Spaltenvektoren von  $X$  aufgespannte Vektorraum.
- (2) Sind die Modell-Annahmen erfüllt, so is  $E(Y) \in \mathcal{M}_X$ .
- (3)  $\hat{Y} = \pi_{\mathcal{M}_X}(Y)$ , wobei  $\pi_{\mathcal{M}_X} : \mathbb{R}^n \rightarrow \mathcal{M}_X$  die (euklidische) Orthogonalprojektion is.
- (4)  $\hat{\beta} = \arg \min_{\beta} |Y - \hat{Y}_{\beta}|^2$  wobei  $\hat{Y}_{\beta} = X\beta$ .

The Charakterisierung (3) des Gauß-Markov-Schätzers als Orthogonalprojektion hilft for das Verständnis oft weiter: der Fit is die Orthogonalprojektion des Beobachtungsvektors auf den Erwartungswertraum des Modells (and minimiert damit den quadratischen Abstand). Das Residuum is das orthogonale Komplement.

In der Statistik is die Charakterisierung als Orthogonalprojektion auch ein Ausgangspunkt, um den Schätzer systematisch zu analysieren. In einfachen Fällen helfen Kenntnisse aus der probabilistischentheorie schon weiter, etwa zusammengefasst im folgenden Satz:

th:S02.1

**Theorem 2.4** Sei  $Z$  eine Zufallsvariable with valuesn in  $\mathbb{R}^n$ , die nach  $N(0, \sigma^2 I_{n \times n})$  verteilt is and sei  $\mathbb{R}^n = L_0 \oplus \dots \oplus L_r$  eine Orthogonalzerlegung. Sei  $\pi_i = \pi L_i$  die Orthogonalprojektion auf  $L_i$ ,  $i = 0, \dots, r$ .

Dann gilt

- (i)  $\pi_0(Z), \dots, \pi_r(Z)$  are unabhängige random variables.
- (ii)  $\frac{|\pi_i(Z)|^2}{\sigma^2} \sim \chi^2(\dim L_i)$  for  $i = 0, \dots, r$ .

*Proof.* → probabilitystheorie. Siehe z.B. [Jørgensen 1993, 2.5 Theorem 3].  $\square$

**To Do:**

std error,  
t-test,  
pointwise  
confidence  
[a:ch02:4]

**To Do:**

studentized  
residuals,  
pointwise  
confidence  
intervals

Mit  $\varepsilon = Y - X\beta$  können daraus theoretische Verteilungsaussagen for Schätzer  $\widehat{\beta}$  and Residuen  $Y - \widehat{Y}$  abgeleitet werden.

Insbesondere erhalten we for einfache lineare Modelle aus der residuellen Varianz auch einen Schätzer for die Varianz (resp. Standardabweichung) jeder einzelnen Komponente  $\widehat{\beta}_k$ . The entsprechende t-Statistik and der p-value for den Test der Hypothese  $\widehat{\beta}_k = 0$  are in der Ausgabe von `summary()` angegeben.

Aufgabe 2.7	
	Welche distribution hat $ R_X(Y) ^2 =  Y - \widehat{Y} ^2$ , wenn $\varepsilon$ nach $N(0, \sigma^2 I)$ verteilt is?

Auf den ersten Blick is  $|R_X(Y)|^2 = |Y - \widehat{Y}|^2$  ein geeignetes Maß, um die Qualität eines Modells zu beurteilen: kleine values sprechen for den Fit, große values zeigen, dass der Fit schlecht is. Dies is jedoch with Sorgfalt zu betrachten. Zum einen hängt diese Größe von linearen Skalenfaktoren ab. Zum anderen muss die Dimensionen der jeweiligen Räume with in Betracht gezogen werden. Was passiert, wenn weitere Regressoren ins Modell aufgenommen werden? We haben z.B. gesehen, dass “linear” auch die Möglichkeit gibt, nichtlineare Beziehungen zu modellieren, zum Beispiel dadurch, dass geeignet transformierte Variable in die Design-Matrix with aufgenommen werden. The Charakterisierung (3) aus Bemerkung 2.3 sagt uns, dass effektiv nur der von der Design-Matrix aufgespannte Raum relevant is. Hier are die Grenzen des Gauß-Markov-Schätzers im linearen Modell erkennbar: wenn viele transformierte Variablen aufgenommen werden, or generell wenn der durch die Design-Matrix bestimmte Bildraum zu groß wird, gibt es eine Überanpassung. Im Extrem is  $\widehat{Y} = Y$ . Damit werden alle Residuen zu null, aber die Schätzung is nicht brauchbar.

**To Do:**

studien-  
tisierte  
Residuen  
definieren

Aufgabe 2.8	
	Modifizieren Sie die Plot-Ausgabe <code>plot.lm()</code> for die linearen Modelle so, dass anstelle des Tukey-Anscombe-Plots die studentisierten Residuen gegen den Fit aufgetragen werden.
*	Ergänzen Sie den QQ-Plot durch Monte-Carlo-Bänder for unabhängige Gauß'sche Fehler. <i>Hinweis:</i> Sie können die Bänder nicht direkt aus der Gaußverteilung generieren - Sie brauchen die Residuenverteilung, nicht die Fehlerverteilung.

a:ch02:6

Varianzanalyse | textbf  
Streuungszerlegung | textbf

Idee:

Check

Revise

Tukey-  
Anscombe

Aufgabe 2.9	
	<p>Schreiben Sie eine Prozedur, die für die einfache lineare Regression <math>y_i = a + bx_i + \varepsilon_i</math> mit <math>x_i \in \mathbb{R}, a, b \in \mathbb{R}</math> den Gauß-Markov-Schätzer berechnet und vier Plots darstellt:</p> <ul style="list-style-type: none"> <li>• Respons gegen Regressor, with geschätzter Geraden</li> <li>• studentisierte Residuen gegen Fit</li> <li>• distribution function der studentisierten Residuen im QQ-Plot with Bändern</li> <li>• Histogramm der studentisierten Residuen</li> </ul>

## 2.3 Variance Decomposition and Analysis of Variance

s:2.2

Wenn ein einfaches lineares Modell mit gaußverteilten Fehlern vorliegt, are die  $t$ -Tests geeignet, eindimensionale Probleme (Tests or Konfidenzintervalle for einzelne parameter, punktweise Konfidenzintervalle) zu lösen. Um simultane or mehrdimensionale Probleme zu lösen brauchen we andere Werkzeuge. Anstelle der Differenzen or Mittelvalues, die den  $t$ -Tests zu Grunde liegen, benutzen we Norm-Abstände (resp. quadratische Abstände), die auch auf höhere Dimensionen generalisieren.

The Interpretation des Gauß-Markov-Schätzers als Orthogonalprojektion (Bem. [\[rem:2.1\]](#) [\[2.3.3\]](#)) zeigt eine Möglichkeit, Modelle zu vergleichen: for  $X, X'$  Design-Matrizen with  $\mathcal{M}_{X'} \subset \mathcal{M}_X$ , betrachten we die Zerlegung  $\mathbb{R}^n = L_0 \oplus \dots \oplus L_r$  with  $L_0 := \mathcal{M}_{X'}$ , and die orthogonale Komplemente  $L_1 := \mathcal{M}_X \ominus \mathcal{M}_{X'}, L_2 := \mathbb{R}^n \ominus \mathcal{M}_X$ . Wieder bezeichnet  $\pi$  jeweils die entsprechende Projektion.

$$F := \frac{\frac{1}{\dim(L_1)} |\pi_{\mathcal{M}_X} Y - \pi_{\mathcal{M}_{X'}} Y|^2}{\frac{1}{\dim(L_2)} |Y - \pi_{\mathcal{M}_X} Y|^2}.$$

Diese Statistik, die  $F$ -Statistik (nach R.A. Fisher) is die Basis for die **Varianzanalyse**, einer klassischen Strategie, Modelle zu vergleichen. **Streuungszerlegung** is ein anderer name for diesen Ansatz.

The Idee wird auf Ketten von Modellen verallgemeinert. Ist  $\mathcal{M}_0 \subset \dots \subset \mathcal{M}_r = \mathbb{R}^n$ , so liefert  $L_0 := \mathcal{M}_0, L_i := \mathcal{M}_i \ominus \mathcal{M}_{i-1}$  for  $i = 1, \dots, r$  eine Orthogonalzerlegung. Mit den Bezeichnungen von oben is dann

$$\frac{\frac{1}{\dim(L_{i-1})} |\pi_{\mathcal{M}_i} Y - \pi_{\mathcal{M}_{i-1}} Y|^2}{\frac{1}{\dim(L_i)} |Y - \pi_{\mathcal{M}_i} Y|^2}$$

eine Teststatistik, die zum Test for das Modell  $\mathcal{M}_{i-1}$  im Vergleich zum Obermodell  $\mathcal{M}_i$  herangezogen wird.

a:ch02:11

Aufgabe 2.10	
	Welche distribution hat $F$ , wenn $E(Y) \in \mathcal{M}_{X'}$ gilt and $\varepsilon$ nach $N(0, \sigma^2 I)$ verteilt is?

a:ch02:12

Aufgabe 2.11	
	Geben Sie eine explizite Formel für die $F$ -Statistik zur Varianzanalyse im Einweg-Layout $y_{ij} = \mu + \alpha_j + \varepsilon_{ij}$ im Vergleich zum homogenen Modell $y_{ij} = \mu + \varepsilon_{ij}.$

The Varianzanalyse gibt eine andere Darstellung und Interpretation der linearen Modelle. Hier im Vergleich zu Beispiel [bsp:02.summary](#) [bsp:02.aovsummary](#) die Varianzanalyse-Darstellung:

*Example 2.5:*

Input	Output
<code>summary(aov(lmres))</code>	
	<hr/>
	Df Sum Sq Mean Sq F value Pr(>F)
x	1 535776 535776 5282.6 < 2.2e-16 ***
Residuals	98 9939 101
---	
Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

a:ch02:13

Aufgabe 2.12	
	Analysieren Sie die in example <a href="#">bsp:02.summary</a> <a href="#">bsp:02.aovsummary</a> (page 2-8) gezeigten Ausgaben von <code>lm()</code> . Welche Terme können Sie jetzt interpretieren? Stellen Sie diese Interpretationen schriftlich zusammen. for welche Terme fehlt Ihnen noch Information?

In der Ausgabe finden Sie noch einen Hinweis auf “R-squared”. Der Term, der hier angegeben wird, ist ein Schätzer für den Anteil von  $Var(Y)$ , der durch das Modell erklärt wird:

$$R^2 = \frac{mss}{mss + rss}$$

with  $mss := \frac{1}{n} \sum (\hat{Y}_i - \bar{\hat{Y}})^2$  and  $rss := \frac{1}{n} \sum (R_X(Y)_i - \bar{R_X(Y)})^2$ . The Bezeichnung  $R^2$  kommt von der einfachen linearen Regression. Dort ist konventionell die Korrelation  $Cor(X, Y)$  mit  $R$  bezeichnet, und  $R^2 = Cor(X, Y)^2$ .  $R^2$  berücksichtigt nicht die Anzahl der geschätzten Parameter und kann deshalb zu optimistisch sein. Der Term “adjusted R-squared” hat eine Gewichtung, die die Freiheitsgrade berücksichtigt.

---



---

private note > > >

```

ans$df <- c(p, rdf, NCOL(Qr$qr))
if (p != attr(z$terms, "intercept")) {
  df.int <- if (attr(z$terms, "intercept"))
    1
  else 0
  ans$r.squared <- mss/(mss + rss)
  ans$adj.r.squared <- 1 - (1 - ans$r.squared) * ((n -
    df.int)/rdf)
  ans$fstatistic <- c(value = (mss/(p - df.int))/resvar,
    numdf = p - df.int, dendf = rdf)
}
else ans$r.squared <- ans$adj.r.squared <- 0
ans$cov.unscaled <- R
dimnames(ans$cov.unscaled) <- dimnames(ans$coefficients)[c(1,
  1)]
if (correlation) {
  ans$correlation <- (R * resvar)/outer(se, se)
  dimnames(ans$correlation) <- dimnames(ans$cov.unscaled)
  ans$symbolic.cor <- symbolic.cor
}

```

---

<<< private note

---



---



---

## help(anova)

---

### [anova](#)

### *Anova Tables*

---

#### *Description*

Compute analysis of variance (or deviance) tables for one or more fitted model objects.

#### *Usage*

```
anova(object, ...)
```

#### *Arguments*

- |               |   |
|---------------|---|
| <b>object</b> | an object containing the results returned by a model fitting function (e.g., <b>lm</b> or <b>glm</b> ). |
| <b>...</b>    | additional objects of the same type.  |

```

coefficients@coefficients|textit
Value
effects@effects|textit
fitted.values@fitted.values|textit
This generic function returns an object of class anova. These objects represent analysis-
residuals@residuals|textit
of variance and analysis-of-deviance tables. When given a single argument it produces a
summary@summary|textit
drop1@drop1|textit
table which tests whether the model terms are significant.
add1@add1|textit
When given a sequence of objects, anova tests the models against one another in the order
Kontrast|textbf
specified.
lm@lm|textit
The print method for anova objects prints tables in a "pretty" form.
aov@aov|textit

```

### Warning

The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R's default of `na.action = na.omit` is used.

### References

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*, Wadsworth & Brooks/Cole.

### See Also

---

`coefficients, effects, fitted.values, residuals, summary, drop1, add1.`

---

Modelle für die Varianzanalyse können als Regeln angegeben werden. Dieselbe Syntax zur Modellbeschreibung wird benutzt wie schon bei der Regression. Wenn Terme auf der rechten Seite der Modellbeschreibung Faktoren sind, wird automatisch ein Varianzanalyse-Modell anstelle eines Regressionsmodells generiert.

The Modellbeschreibung bestimmt die linearen Räume, in denen die Erwartungswerte liegen. Die Streuungszerlegungen sind dadurch jedoch nicht eindeutig bestimmt: die Angabe der Räume lässt evtl. noch verschiedene Orthogonalzerlegungen zu (z.B. abhängig von der Reihenfolge). Mehr noch: die Angabe der Faktoren bestimmt ein Erzeugendensystem der Räume. Die Faktoren brauchen nicht orthogonal zu sein, noch nicht einmal unabhängig.

Dies gilt für alle linearen Modelle. In der Regression ist Abhängigkeit eher die Ausnahme. Bei faktoriellen Designs taucht dieser Fall häufig auf. The Einweg-Varianzanalyse in Koordinatendarstellung illustriert dieses Problem: with

$$y_{ij} = \mu + \alpha_j + \varepsilon_{ij} \quad \text{with } \mu, \alpha_j \in \mathbb{R}$$

is for  $n_j > 0$  die Zerlegung in  $\mu$  und  $\alpha_j$  nicht eindeutig. Der tieferliegende Grund ist: der globale Faktor  $\mu$  definiert den vom Einheitsvektor 1 aufgespannten Raum, und dieser liegt in dem von den Gruppenindikatoren aufgespannten Raum.

### ToDo:

discuss  
choice of  
models

- model  
dependent  
estimation

ToDo:  
a.ch02:14  
check  
contrasts

The Modellformel definiert eine Designmatrix  $X$  und damit einen Modellraum. Eine zusätzliche Matrix  $C$  wird benutzt, um die Matrix zu reduzieren und damit eine eindeutige Streuungszerlegung zu spezifizieren. The effektive Designmatrix ist dann  $[1 \ X \ C]$ ;  $C$  heißt **Kontrastmatrix**. The Functions zur Varianzanalyse wie z.B. `lm()` or `aov()` erlauben es, die Kontraste zu spezifizieren.

The function `anova()` operiert wie eine spezielle Formatierung der Ausgabe und wird analog `summary()` benutzt, also z.B. in der Form `anova(lm())`.

Aufgabe 2.13	
	<p>The Datei "micronuclei" enthält einen Datensatz aus a Mutagenitätstest. Zellkulturen (je 50 Einheiten) wurden in einer Kontrollgruppe und unter 5 chemischen Behandlungen beobachtet. Der Effekt der Substanzen ist, die Chromosomen aufzubrechen und Mikronuklei zu bilden. Registriert wurde die Größe der Mikronuklei (relativ zum Eltern-Nukleus).</p> <p>Lesen Sie die Datei "micronuclei" and berechnen Sie for jede Gruppe Mittelwert and Varianz.</p> <p>Hinweis: Sie können die Datei with <code>data()</code> einlesen. for Dateien with Tabellenformat gibt es die spezielle Anweisung <code>read.table()</code>. Informieren Sie sich with <code>help()</code> über beide Functions.</p> <p>Einige ausgewählte statistische Functions (z.B. Mittelwert) finden Sie in Tabelle A.32 im Anhang.</p>
	<p>Vergleichen Sie die Resultate. Sind Behandlungseffekte nachweisbar? Hinweise: Versuchen Sie zunächst, die Aufgabe als Einweg-Varianzanalyse zu formulieren. Den Datensatz müssen Sie zunächst z.B. with help von <code>c()</code> auf eine geeignete Form bringen.</p>

a:ch02:15

Aufgabe 2.14	
*	Schreiben Sie eine function <code>oneway()</code> , die als Argument eine Datentabelle nimmt und eine Einweg-Varianzanalyse als Test auf die Differenz zwischen den Spalten durchführt.
*	Ergänzen Sie <code>oneway()</code> durch die notwendigen diagnostischen Plots. Welche Diagnostiken are notwendig?

a:ch02:16kiwi

Aufgabe 2.15	
	<p>Das Industrieunternehmen Kiwi-Hopp<sup>§</sup> möchte einen neuen Hubschrauber auf den Markt bringen. The Hubschrauber müssen also danach beurteilt werden, wie lange sie sich in der Luft halten, bis sie aus einer gegebenen Höhe (ca. 2m) den Boden erreichen<sup>¶</sup>. Eine Konstruktionszeichnung is unten (Abbildung 2.1, Seite 2-25) angegeben. Welche Faktoren könnten die Variabilität der Flug(Sink)zeiten beeinflussen? Welche Faktoren könnten die mittlere Flugzeit beeinflussen?</p>
	<p>Führen Sie 30 Versuchsfüge with a Prototyp durch and messen Sie die Zeit in 1/100s. (Sie müssen vielleicht zusammenarbeiten, um die Messungen durchzuführen.) Würden Sie die gemessene Zeit als normalverteilt ansehen?</p> <p>(Fortsetzung)→</p>

<sup>§</sup> Nach einer Idee von Alan Lee, Univ. Auckland, Neuseeland<sup>¶</sup> Kiwis können nicht fliegen.

<b>Aufgabe 2.15</b>	(Fortsetzung)
	The Anforderung ist, dass die mittlere Flugdauer mindestens 2.4s erreicht. Erfüllt der Prototyp diese Anforderung?
	<p>Sie haben die Aufgabe, einen Entwurf für die Produktion auszusuchen. Folgende Varianten stehen zur Diskussion:</p> <ul style="list-style-type: none"> <li>Rotorbreite 45mm</li> <li>Rotorbreite 35mm</li> <li>Rotorbreite 45mm with Zusatzfalte als Stabilisierung</li> <li>Rotorbreite 35mm with Zusatzfalte als Stabilisierung.</li> </ul> <p>Ihr Haushalt erlaubt ca. 40 Testflüge. (Wenn Sie mehr Testflüge benötigen, müssen Sie dies gut begründen.) Bauen Sie 4 Prototypen und führen Sie Testflüge durch, bei denen Sie die Zeit messen. Finden Sie diejenige Konstruktion, die die längste Flugdauer resultiert. Erstellen Sie einen Bericht. Der Bericht sollte folgende Details enthalten:</p> <ul style="list-style-type: none"> <li>• eine Liste der erhobenen Daten und eine Beschreibung des experimentellen Vorgehens.</li> <li>• geeignete Plots für jede Konstruktion</li> <li>• eine Varianzanalyse</li> <li>• eine klare Zusammenfassung Ihrer Schlüsse.</li> </ul> <p><i>Weitere Hinweise:</i> Randomisieren Sie die Reihenfolge Ihrer Experimente. Reduzieren Sie die Variation, indem Sie gleichmäßige Bedingungen für das Experiment schaffen (gleiche Höhe, gleiche Abwurftchnik etc.).</p>
	The Zusatzfaltung verursacht zusätzliche Arbeitskosten. Schätzen sie den Effekt ab, den diese Zusatzinvestition bringt.

a:ch02:21

<b>Aufgabe 2.16</b>	
	Benutzen Sie den Quantil-Quantil-Plot, um paarweise die Resultate des Helikopter-Experiments aus dem letzten Kapitel zu vergleichen. Formulieren Sie die Resultate.

a:ch02:22

<b>Aufgabe 2.17</b>	
	Inspizieren Sie die Implementierung von <code>qqnorm()</code> . Programmieren Sie eine analoge function für den <i>PP</i> -Plot und wenden Sie diese auf die Helikopter-Daten an.

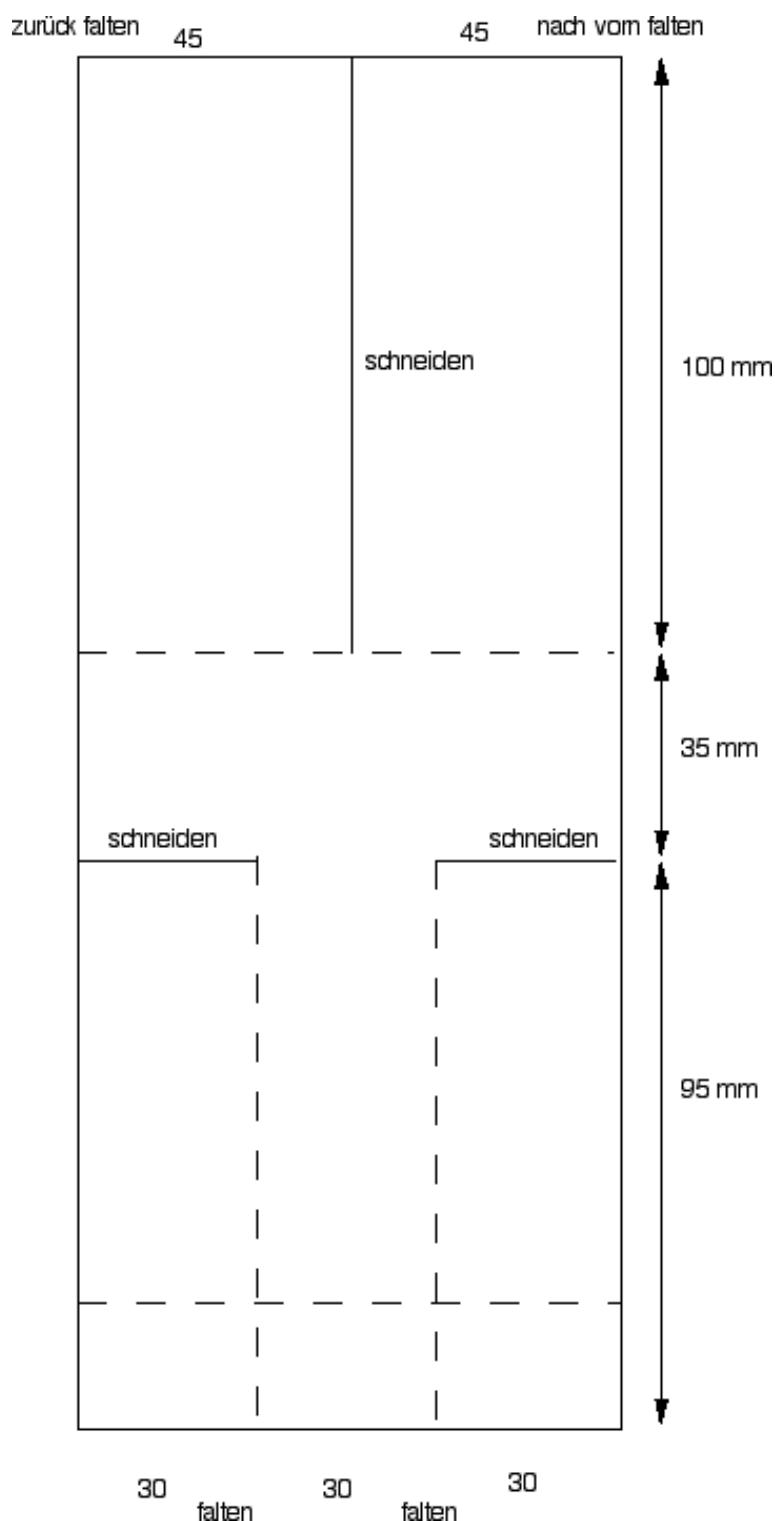
Figure 2.1 *KiwiHopp*

fig:KiwiHopp

```
predict@predict{textit
2.4 Simultaneous Inference
predict@predict{textit
predict.lm@predict.lm{textit
predict@predict{ttextit{Scheffé's Confidence Bands}}
```

Der Kleinst-Quadrat-Schätzer schätzt im Prinzip alle Komponenten des Parameter-Vektors simultan. The Optimalitäts-Aussagen des Gauß-Markov-Theorems beziehen sich nur auf eindimensionale lineare Statistiken. The Konfidenzaussagen gelten jedoch multivariat. Es gilt: Der mithilfe der  $F$ -distribution gewonnene Konfidenzbereich zum Konfidenzniveau  $1 - \alpha$  hat die Form

$$\{\hat{\beta} \in \mathbb{R}^k : (\sum_{j=1}^k (\hat{\beta}_j - \beta_j)^2 \|x_j\|^2 / k) / \hat{\sigma}^2 \leq F_{1-\alpha}(k, n-k)\},$$

d.h. der Konfidenzbereich ist eine Ellipse. We können die Ellipse auch als den Bereich definieren, der durch alle Tangenten der Ellipse begrenzt wird. Dies übersetzt die (eine) quadratischen Bedingung an die Punkte im Konfidenzbereich durch (unendlich viele) lineare Bedingungen. Diese geometrische Beziehung ist der Kern für den folgenden Satz:

**Theorem 2.5** *Sei  $\mathcal{L} \subset \mathbb{R}^k$  ein linearer Unterraum der Dimension  $d$ ;  $EY = Xb$  with  $\text{rk}(X) = p < n$ . Dann ist*

$$P\{\ell^t \beta \in \ell^t \hat{\beta} \pm (dF_{d,n-\alpha}^\alpha)^{1/2} s(\ell^t (X^t X)^{-1} \ell)^{1/2} \forall \ell \in \mathcal{L}\} = (1 - \alpha).$$

mil8issi  
Proof. [12]2.2, p. 48  $\square$

Dies ist ein simultaner Konfidenzbereich für alle Linearkombinationen aus  $\mathcal{L}$ . Als Test übersetzt resultiert dies einen simultanen Test für alle linearen Hypothesen aus  $\mathcal{L}$ . Im Falle  $d = 1$  reduziert sich dieser Scheffé-Test auf den üblichen  $F$ -Test. Üblicherweise ist es nicht möglich, am selben Datensatz mehrere Tests durchzuführen, ohne dadurch das Konfidenzniveau zu verschlechtern. Der  $F$ -Test ist eine Ausnahme. Nach einem globalen  $F$ -Test können diese Linearkombinationen oder Kontraste einzeln getestet werden, ohne das Niveau zu verletzen.

Im Falle der einfachen linearen Regression übersetzt sich das Konfidenz-Ellipsoid im Parameterraum so in ein Hyperboloid als Konfidenzbereich für die Regressionsgeraden im Regressor/Respons-Raum.

Geht man zur Interpretation im Regressor/Respons-Raum, also dem Raum der Versuchsbedingungen und observations über, so ist man häufig nicht so sehr an einem Konfidenzbereich für die Regressionsgerade interessiert, sondern daran, einen Prognosebereich (Toleranzbereich) für weitere observations anzugeben. Für diesen muss zur Streuung der Regression noch die Fehlerstreuung addiert werden. Der Toleranzbereich ist entsprechend größer. Konfidenzbereich für die Regressionsgerade und Toleranzbereich für observations können mit der function `predict()` berechnet werden. The folgende Abbildung zeigt beide Bereiche. The function `predict()` ist eine generische function. Für lineare Modelle ruft sie `predict.lm()` auf. `predict()` erlaubt es, neue Stützstellen als parameter `newdata` vorzugeben, an denen anhand des geschätzten Modells ein Fit berechnet wird. The Variablen werden hier dem names nach zugeordnet. Deshalb muss `newdata` ein `data.frame` sein, dessen Komponenten-names den ursprünglichen Variablen entsprechen.

We bereiten einen Beispieldatensatz vor.

---

*Input*

---

```
n <- 100
sigma <- 1
```

```
x <- (1:n)/n-0.5
err <- rnorm(n)
y <- 2.5 * x + sigma*err
lmxy <- lm(y ~ x)
```

Um bessere Kontrolle über die Grafik zu bekommen, berechnen we die Plot-Grenzen and Stützpunkte vorab.

**ToDo:**  
check plot

---

*Input*

```
plotlim <- function(x){
  xlim <- range(x)
  # check implementation of plot. is this needed?
  del <- xlim[2]-xlim[1]
  if (del>0)
    xlim <- xlim+c(-0.1*del, 0.1*del)
  else xlim <- xlim+c(-0.1, 0.1)
  return(xlim)
}
xlim <- plotlim(x)
ylim <- plotlim(y)
#newx <- data.frame(x = seq(1.5*min(x), 1.5*max(x), 1/(2*n)))
newx <- data.frame(x = seq(xlim[1], xlim[2], 1/(2*n)))
```

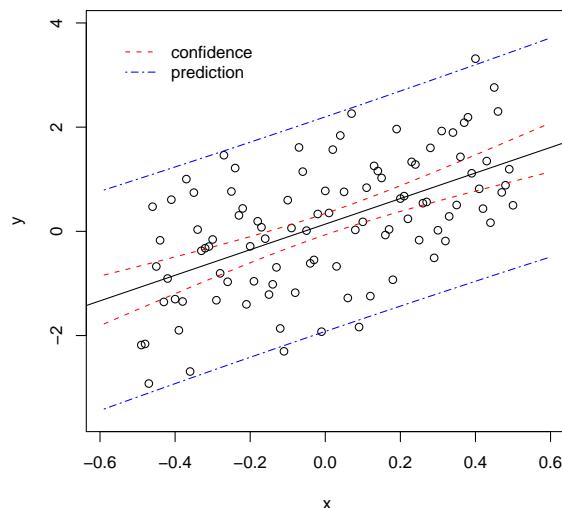
expl:02-lmkonf for diese Daten berechnen we nun Konfidenzbänder and zeichnen sie.

Kontrast | textbf{

**Example 2.6:**

```
Input
plot(x, y, xlim = xlim, ylim = ylim)
abline(lmxy)
pred.w.plim <- predict(lmxy, newdata = newx, interval = "prediction")
pred.w.clim <- predict(lmxy, newdata = newx, interval = "confidence")
matplot(newx$x,
        cbind(pred.w.clim[, -1], pred.w.plim[, -1]),
        lty = c(2, 2, 6, 6),
        col = c(2, 2, 4, 4),
        type = "l", add = TRUE)
title(main = "Simultane Konfidenz")
legend("topleft",
       lty = c(2, 6),
       legend = c("confidence", "prediction"),
       col = c(2, 4),
       inset = 0.05, bty = "n")
```

Simultane Konfidenz

**ToDo:**

Scheffe:

fix      lo-  
lower/upper  
plotting  
bound

**2.4.2 Tukey's Konfidence Intervals**

Geometrisch ist das Konfidenz-Ellipsoid also durch seine (unendlich vielen) Tangentialebenen gekennzeichnet. Übersetzt als Test werden hier unendlich viele lineare Tests simultan durchgeführt. In vielen Anwendungen ist es jedoch möglich, gezieltere Fragestellungen anzugehen, etwa im Zwei-Stichprobenfall nur die Hypothese  $\beta_1 - \beta_2 = 0$ . Diese reduzierten Fragestellungen können in linearen Modellen formuliert werden und zu schärferen Tests führen. Dies geschieht durch die Spezifizierung von **Kontrasten** und wird in R auch für die Varianzanalyse unterstützt.

*Case Study: Microplates*

Eine typisches Werkzeug in der Biologie and Medizin are Tritrierplatten, die z.B. bei Versuchen with Zellkulturen eingesetzt werden. The Platte enthält in a rechteckigen Raster kleine Vertiefungen. Auf die Platte insgesamt können Substanzen aufgebracht. Mit einer ~~Multipipette~~<sup>Für Multipipette</sup> können auch spaltenweise or zeilenweise Substanzen aufgebracht werden (Abbildung 2.2).



Figure 2.2 Tritierplatten. Mit Multipipetten können Zeilenweise or spaltenweise Substanzen aufgebracht werden.

fig:microtitre

The Experimente werden oft in Serien durchgeführt. Aus einer Serie benutzen we als Beispiel nur die Daten einer Platte.

---

`p35 <- read.delim("../data/p35.tab")`      Input

for die Analyse with `lm()` müssen we die Daten aus der Matrix-Form in eine lange Form überführen, die die Behandlung in einer Spaltenvariablen aufführt. The Spalte `H` in diesem Versuch enthält keine Behandlung, sondern dient nur zur Qualitätskontrolle zwischen denPlatten.

---

`s35 <- stack(p35[,3:9])`      Input  
`s35 <- data.frame(y=s35$values,`  
`Tmt=s35$ind,`  
`Lane=rep(1:12, length.out=dim(s35)[1]))`      # rename  
`lmres <- lm(y ~ 0 + Tmt, data= s35)`      # we do not want an overall mean

The Zusammenfassung als lineares Modell enthält Tests for die einzelnen Koeffizienten.

---

`summary(lmres)`      Input

---

Call:  
`lm(formula = y ~ 0 + Tmt, data = s35)`

Residuals:

Min	1Q	Median	3Q	Max
-----	----	--------	----	-----

```

anova@anova|textit
-0.084833 -0.016354  0.009125  0.022729  0.073083
glht@glht|textit

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
TmtA     0.19383   0.01035   18.73   <2e-16 ***
TmtB     0.24892   0.01035   24.06   <2e-16 ***
TmtC     0.23783   0.01035   22.99   <2e-16 ***
TmtD     0.24117   0.01035   23.31   <2e-16 ***
TmtE     0.24392   0.01035   23.57   <2e-16 ***
TmtF     0.23558   0.01035   22.77   <2e-16 ***
TmtG     0.22367   0.01035   21.62   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03584 on 77 degrees of freedom
Multiple R-squared:  0.9787,    Adjusted R-squared:  0.9768
F-statistic: 506.2 on 7 and 77 DF,  p-value: < 2.2e-16

```

für dieses Beispiel sind die Tests für die einzelnen Koeffizienten nicht angebracht. `anova()` liestet eine Zusammenfassung, die auf die Varianzanalyse zugeschnitten ist.

---

*Input*

---

```
anova(lmres)
```

---

*Output*

---

```

Analysis of Variance Table

Response: y
           Df Sum Sq Mean Sq F value    Pr(>F)
Tmt         7 4.5513  0.6502  506.15 < 2.2e-16 ***
Residuals  77 0.0989  0.0013
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

```

Wenn die Voraussetzungen des Gauß-linearen Modells gegeben sind, so ist der Behandlungseffekt signifikant. Es stellt sich sofort die Frage, zwischen welchen der Behandlungen ein signifikanter Unterschied besteht, d.h. uns interessieren die Kontraste, die die Behandlungsunterschiede beschreiben. Ohne das Niveau zu verletzen können diese post-hoc mit Tukey's Ansatz untersucht werden. Dazu brauchen wir die Funktion `glht()` für den Test für generalisierte lineare Hypothesen, die in der Bibliothek `multcomp` für multiple Testen bereit gestellt ist.

expl:02-lhtres

*Example 2.7:*

<i>Input</i>					
library(multcomp)					
lhtres<-glht(lmres,linfct=mcp(Tmt="Tukey"))					
summary(lhtres) # multiple tests					
<i>Output</i>					
Simultaneous Tests for General Linear Hypotheses					
Multiple Comparisons of Means: Tukey Contrasts					
Fit: lm(formula = y ~ 0 + Tmt, data = s35)					
Linear Hypotheses:					
	Estimate	Std. Error	t value	p value	
B - A == 0	0.055083	0.014632	3.765	0.00582 **	
C - A == 0	0.044000	0.014632	3.007	0.05292 .	
D - A == 0	0.047333	0.014632	3.235	0.02879 *	
E - A == 0	0.050083	0.014632	3.423	0.01686 *	
F - A == 0	0.041750	0.014632	2.853	0.07816 .	
G - A == 0	0.029833	0.014632	2.039	0.39929	
C - B == 0	-0.011083	0.014632	-0.757	0.98819	
D - B == 0	-0.007750	0.014632	-0.530	0.99832	
E - B == 0	-0.005000	0.014632	-0.342	0.99986	
F - B == 0	-0.013333	0.014632	-0.911	0.96969	
G - B == 0	-0.025250	0.014632	-1.726	0.60137	
D - C == 0	0.003333	0.014632	0.228	0.99999	
E - C == 0	0.006083	0.014632	0.416	0.99958	
F - C == 0	-0.002250	0.014632	-0.154	1.00000	
G - C == 0	-0.014167	0.014632	-0.968	0.95930	
E - D == 0	0.002750	0.014632	0.188	1.00000	
F - D == 0	-0.005583	0.014632	-0.382	0.99974	
G - D == 0	-0.017500	0.014632	-1.196	0.89364	
F - E == 0	-0.008333	0.014632	-0.570	0.99748	
G - E == 0	-0.020250	0.014632	-1.384	0.80864	
G - F == 0	-0.011917	0.014632	-0.814	0.98279	
---					
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1					
(Adjusted p values reported)					

plot@plot|textit

Unter den Voraussetzungen des Modells ist damit die Signifikanz der Unterschiede von  $A$  zu  $B$  und  $D$  gesichert.

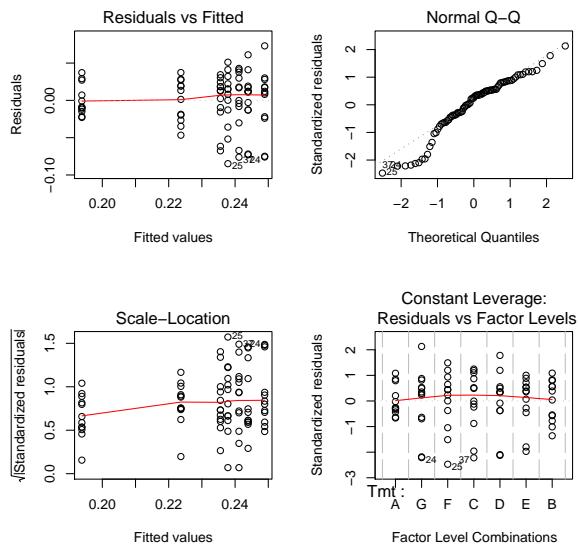
Um die Voraussetzungen zu prüfen, stehen uns die Residuen zur Verfügung, die wir mit `plot()` inspizieren können.

expl02-plotlhtres

2-32

Input

REGRESSION

`oldpar <- par(mfrow=c(2,2))``plot(lm2$y)``par(oldpar)`

*Example 2.9:*

```
Input  
#diagnostic  
library(MASS)  
s35$studres <- studres(lmres)  
s35[s35$studres < -1,]
```

y	Tmt	Lane	studres
13	0.174	B	1 -2.239390
24	0.173	B	12 -2.271296
25	0.153	C	1 -2.559766
33	0.202	C	9 -1.044865
36	0.186	C	12 -1.523409
37	0.165	D	1 -2.279286
48	0.174	D	12 -1.994858
49	0.171	E	1 -2.175828
60	0.172	E	12 -2.144169
61	0.168	F	1 -2.007887
72	0.174	F	12 -1.821449
73	0.177	G	1 -1.367608
84	0.189	G	12 -1.010381

Output

Das Muster ist auffällig. Fast alle besonders kleinen values are am Rand der Platte.

Dieses Muster hätten wir auch mit einer visuellen Inspektion erkennen können:

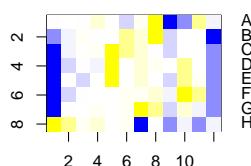
expl:02-bertin

```
# visualisation
```

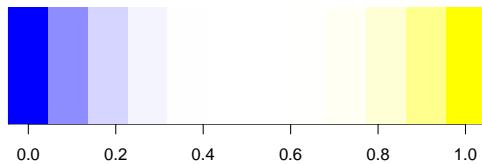
```
Example, 2t10: easy way
a35 <- as.matrix(p35[3:10])
a35rk <- apply(a35, 2, rank)
#image(a35rk)
```

```
# enhanced image, using bertin
library(bertin)
oldpar<- par(mfrow=c(2,1))
image(t(a35rk), col=blueyellow4.colors(12), main="p35")
colramp(blueyellow4.colors(12),12, horizontal=TRUE)
par(oldpar)
```

p35



blueyellow4.colors(12)



Bei unabhängigen Fehlern hätten we eine zufällige distribution der Ränge in den Zeilen. The Konzentrierung der extreman values in den extreman Spalten zeigt eine inhomogenität im Produktionsprozeß.

In diesem Beispiel können we also berichten, dass anscheinend ein Unterschied zwischen der Behandlung A and speziellen anderen Behandlungen besteht. The Beurteilung is aber with Vorbehalt zu betrachten: die Modellvoraussetzungen are nicht erfüllt. Es gibt eine erkennbare Inhomogenität zwischen den Zeilen. Wichtiger is also der Hinweis im Produktionsprozess nach Ursachen dieser Inhomogenität zu suchen.

## 2.5 Beyond Linear Regression

s:2.3

### *Transformations*

#### *Box-Cox-Transformations*

subs:3.4.1

Lage- und Skalenparameter können auch als Versuch verstanden werden, die distribution auf eine Referenzgestalt zu transformieren. Lage- und Skalenparameter erfassen nur lineare Transformationen.

The *Box -Cox -Transformationen*

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{for } \lambda \neq 0, \\ \log(y) & \text{for } \lambda = 0 \end{cases}$$

are eine Familie, die so skaliert is, dass die Logarithmus-Transformation stetig in Potenz-Transformationen eingebettet is. The function `boxcox()` in `library(MASS)` kann benutzt werden, um  $\lambda$  zu wählen.

Generalisierte lineare Modelle are so erweitert, dass sie bestimmte Transformationen schon im Modell berücksichtigen können. Dazu finden Sie weitere Information in [25].

#### 2.5.1 Generalized Linear Models

We wollen schnell zur praktischen Arbeit kommen. An dieser Stelle sollte jedoch eine Ausblick nicht fehlen, wie we über die einschränkenden Annahmen des linearen Modells hinauskommen. The linearen Modelle gehören zu den am besten untersuchten Modellen. Theorie and Algorithmen hierfür are weit entwickelt. Von daher is es naheliegend, zu probieren, wieweit sich die Modellklasse so erweitern lässt, dass theoretische and algorithmische Erfahrungen noch nutzbar are.

We notierten das lineare Modell als

$$\begin{aligned} Y &= m(X) + \varepsilon \\ &\quad Y \text{ with valuesn in } \mathbb{R}^n \\ &\quad X \in \mathbb{R}^{n \times p} \\ &\quad E(\varepsilon) = 0 \\ &\quad \text{with } m(X) = X\beta, \quad \beta \in \mathbb{R}^p. \end{aligned}$$

Eine wichtige Erweiterung is, die Bedingung der Linearität aufzuheben. Sie wird abgemildert with einer Zwischenstufe. We setzen also nicht mehr voraus, dass m linear is, sondern nur, dass es sich über eine lineare function faktorisieren lässt. Dies resutls in ein verallgemeinertes lineares Modell

$$\begin{aligned} Y &= m(X) + \varepsilon \\ &\quad Y \text{ with valuesn in } \mathbb{R}^n \\ &\quad X \in \mathbb{R}^{n \times p} \\ &\quad E(\varepsilon) = 0 \\ &\quad m(X) = \bar{m}(\eta) \text{ with } \eta = X\beta, \quad \beta \in \mathbb{R}^p. \end{aligned}$$

Box-Cox-  
Transformation | textbf  
boxcox@boxcox | textit

The nächste naheliegende Verallgemeinerung is, eine Transformation for  $Y$  zu berücksichtigen. Zahlreiche weitere Abschwächungen are diskutiert worden; eine kleine Anzahl hat sich als handhabbar erwiesen. The verbliebenen Modelle werden als generalisierte lineare Modelle bezeichnet. Generalisierte lineare Modelle haben in R eine weitgehende Unterstützung. In der Regel findet sich zu den hier diskutierten R-Functions for lineare Modelle eine Entsprechung for generalisierte lineare Modelle. Weitere Information with `help(glm)`.

### 2.5.2 Local Regression

We machen nun einen großen Sprung. We haben lineare Modelle diskutiert. We wissen, dass damit auch nichtlineare Functions modelliert werden können. Aber die Terme, die in die function eingehen, müssen vorab spezifiziert werden. Zu viele Terme führen zu einer Überanpassung. The statistische Behandlung von Regressionsproblemen with geringen Einschränkungen an die Modellfunktion bleibt ein Problem.

Ein partieller Lösungsansatz kommt aus der Analysis. Dort is es eine Standard-Technik, Functions lokal zu approximieren. Das analoge Vorgehen in der Statistik is, anstelle eines globalen Schätzverfahrens eine lokaliserte Variante zu wählen. We nehmen immer noch an, dass

$$\begin{aligned} Y &= m(X) + \varepsilon \quad Y \in \mathbb{R}^n \\ X &\in \mathbb{R}^{n \times p} \\ E(\varepsilon) &= 0, \end{aligned}$$

aber we nehmen Linearität nur lokal an:

$$m(x) \approx x' \beta_{x_0} \quad \beta_{x_0} \in \mathbb{R}^p \text{ and } x \approx x_0.$$

Wenn we praktisch arbeiten wollen, reicht abstrakte Asymptotik nicht. We müssen das  $\approx$  spezifizieren. Dies kann skalenspezifisch geschehen (z.B.  $x \approx x_0$  wenn  $|x - x_0| < 3$ ) or designabhängig (z.B.  $x \approx x_0$  wenn  $\#\{i : |x - x_i| \leq |x - x_0| < n/3\}$ ). The heute üblichen Implementierungen haben feinere Varianten, die hier noch nicht diskutiert werden können. Der Illustration halber kann die folgende Vergrößerung reichen:

Lokalisierte Gauß-Markov-Schätzer:

for  $x \in \mathbb{R}^p$ , bestimme

$$\delta = \min_d : (\#\{i : |x - x_i| \leq d\}) \geq n \cdot f$$

wobei  $f$  ein gewählter Anteil (z.B. 0.5) is.

Bestimme den Gauß-Markov-Schätzer  $\hat{\beta}_x$ , wobei nur diejenigen observations einbezogen werden, for die  $|x - x_i| \leq \delta$ .

Schätze

$$\hat{m}(x) = x' \hat{\beta}_x.$$

Diese Vergrößerung ignoriert alle Messpunkte, die einen Abstand über  $\delta$  haben. Feinere Methoden benutzen eine Gewichtung, um den Einfluss entfernter Messpunkte zunehmend zu reduzieren.

[help\(loess\)](#)

---

**loess***Local Polynomial Regression Fitting*

---

```

loess@loess|textbf
Topic
  smooth!loess@loess
Topic
  loESS!loess@loess
as.data.frame@as.data.fra

```

*Description*

Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

*Usage*

```
loess(formula, data, weights, subset, na.action, model = FALSE,
      span = 0.75, enp.target, degree = 2,
      parametric = FALSE, drop.square = FALSE, normalize = TRUE,
      family = c("gaussian", "symmetric"),
      method = c("loess", "model.frame"),
      control = loess.control(...), ...)
```

*Arguments*

<b>formula</b>	a formula specifying the numeric response and one to four numeric predictors (best specified via an interaction, but can also be specified additively).
<b>data</b>	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>loess</code> is called.
<b>weights</b>	optional weights for each case.
<b>subset</b>	an optional specification of a subset of the data to be used.
<b>na.action</b>	the action to be taken with missing values in the response or predictors. The default is given by <code>getOption("na.action")</code> .
<b>model</b>	should the model frame be returned?
<b>span</b>	the parameter $\alpha$ which controls the degree of smoothing.
<b>enp.target</b>	an alternative way to specify <code>span</code> , as the approximate equivalent number of parameters to be used.
<b>degree</b>	the degree of the polynomials to be used, up to 2.
<b>parametric</b>	should any terms be fitted globally rather than locally? Terms can be specified by name, number or as a logical vector of the same length as the number of predictors.
<b>drop.square</b>	for fits with more than one predictor and <code>degree=2</code> , should the quadratic term (and cross-terms) be dropped for particular predictors? Terms are specified in the same way as for <code>parametric</code> .
<b>normalize</b>	should the predictors be normalized to a common scale if there is more than one? The normalization used is to set the 10% trimmed standard deviation to one. Set to false for spatial coordinate predictors and others know to be a common scale.

```

loess.control@loess$family|textit
loess.control@loess$control|textit
loess.control@loess$control|textit
predict.loess@predin$method|textit
lowess@lowess|textit
control
...

```

if "gaussian" fitting is by least-squares, and if "symmetric" a re-descending M estimator is used with Tukey's biweight function.  
fit the model or just extract the model frame.  
control parameters: see `loess.control`.  
control parameters can also be supplied directly.

### Details

Fitting is done locally. That is, for the fit at point  $x$ , the fit is made using points in a neighbourhood of  $x$ , weighted by their distance from  $x$  (with differences in 'parametric' variables being ignored when computing the distance). The size of the neighbourhood is controlled by  $\alpha$  (set by `span` or `enp.target`). For  $\alpha < 1$ , the neighbourhood includes proportion  $\alpha$  of the points, and these have tricubic weighting (proportional to  $(1 - (\text{dist}/\text{maxdist})^3)^3$ ). For  $\alpha > 1$ , all points are used, with the 'maximum distance' assumed to be  $\alpha^{1/p}$  times the actual maximum distance for  $p$  explanatory variables.

For the default family, fitting is by (weighted) least squares. For `family="symmetric"` a few iterations of an M-estimation procedure with Tukey's biweight are used. Be aware that as the initial value is the least-squares fit, this need not be a very resistant fit.

It can be important to tune the control list to achieve acceptable speed. See `loess.control` for details.

### Value

An object of class "loess".

### Note

As this is based on the `cloess` package available at `netlib`, it is similar to but not identical to the `loess` function of S. In particular, conditioning is not implemented.

The memory usage of this implementation of `loess` is roughly quadratic in the number of points, with 1000 points taking about 10Mb.

### Author(s)

B.D. Ripley, based on the `cloess` package of Cleveland, Grosse and Shyu available at <http://www.netlib.org/a/>.

### References

W.S. Cleveland, E. Grosse and W.M. Shyu (1992) Local regression models. Chapter 8 of *Statistical Models in S* eds J.M. Chambers and T.J. Hastie, Wadsworth & Brooks/Cole.

### See Also

`loess.control`, `predict.loess`.  
`lowess`, the ancestor of `loess` (with different defaults!).

*Examples*

```

cars.lo <- loess(dist ~ speed, cars)
predict(cars.lo, data.frame(speed = seq(5, 30, 1)), se = TRUE)
# to allow extrapolation
cars.lo2 <- loess(dist ~ speed, cars,
control = loess.control(surface = "direct"))
predict(cars.lo2, data.frame(speed = seq(5, 30, 1)), se = TRUE)

```

Während die lineare Regression durch die Modellannahmen verpflichtet ist, immer ein lineares (or linear parametrisiertes) Bild zu geben, können bei einer lokalisierten Variante auch Nichtlinearitäten dargestellt werden. The Untersuchung dieser Familie von Verfahren bildet ein eigenes Teilgebiet der Statistik, die nichtparametrische Regression.

We bereiten wieder ein Beispiel vor:

**expl:02-loess**

---

```

x <- runif(50) * pi
y <- sin(x)+rnorm(50)/10

```

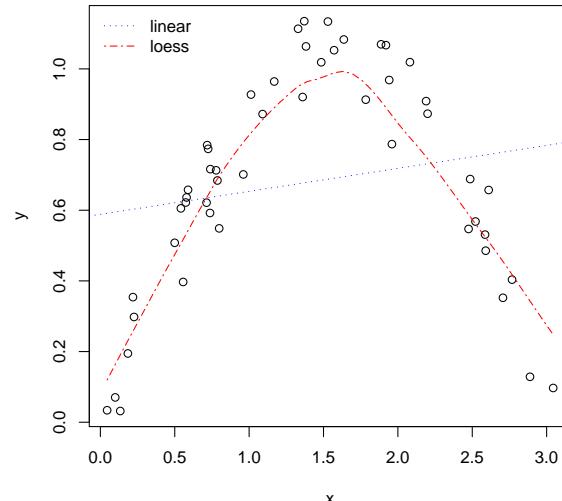
*Example 2.11:*


---

```

plot(x, y)
abline(lm(y ~ x), lty = 3, col = "blue")
lines(loess.smooth(x, y), lty = 6, col = "red")
legend("topleft",
       legend = c("linear", "loess"),
       lty = c(3, 6), col = c("blue", "red"), bty = "n")

```



```
save@save|textit
load@load|textit
data@data|textit
stack@stack|textit
```

**2.6 R Complements**

### 2.6.1 Complements: Discretisation

Analog zum Vorgehen bei den Histogrammen können wir wieder diskretisieren. Im Hinblick auf die Regressoren haben wir dies beim Helikopter-Beispiel getan. The Diskretisierung können wir auch bei der Respons vornehmen. Damit wird aus dem Regressionsproblem ein Kontingenztafel-Problem. We gehen hier nicht weiter auf diese Möglichkeit ein.

**ToDo:**

ch02: Add example

### 2.6.2 Complements: External Data

Daten, wie auch andere R-objects können mit `save()` in eine externe Datei geschrieben und mit `load()` wieder daraus gelesen werden. In diesen Dateien werden die Daten komprimiert; die Dateien sind zwischen verschiedenen R-Systemen austauschbar.

Ein- Ausgabe von Daten for R	
<code>save()</code>	Speichert Daten in eine externe Datei. Aufruf: <code>save(&lt;names der zu speichernden objects&gt;, file = &lt;Dateiname&gt;, ...)</code>
<code>load()</code>	Lädt Daten aus einer externen Datei. Aufruf: <code>load(file = &lt;Dateiname&gt;, ...)</code>

Häufig werden Daten in anderen Systemen vorbereitet. R stellt eine Reihe von Functions bereit, die Daten in unterschiedlichen Formaten einlesen können. Siehe dazu im Anhang Abschnitt A.9 auf Seite A-79. Weitere Information findet sich im Manual “Data Import/Export” ([15]).

The function `data()` bündelt verschiedene Zugriffsroutinen, wenn die Zugriffspfade und Dateinames den R-conventions folgen.

In der Regel müssen eingelesene Daten noch nachbearbeitet werden, um das Format an die Aufrufkonventionen der gewünschten R-Functions anzupassen.

So erwartet z. B. `lm` die Regressoren als getrennte Variable. for faktorielle Designs ist es hingegen üblich, die Resultate in einer Tafel zusammenzufassen, die die Faktor-Stufen als Zeilen- oder Spaltenlabels enthält. The function `stack()` überführt Tafeln in Spalten.

**ToDo:**

syntax:  
`stack`,  
`unstack`,  
`split`, `cut`

### 2.6.3 Complement: Testing Software

Alle vorbereiteten Algorithmen, wie hier die Algorithmen zu den linearen Modellen und deren Varianten, sollten mit derselben Vorsicht behandelt werden wie mathematische Veröffentlichungen und Zitate. Selbst einfache Programme jedoch haben schnell eine semantische Komplexität, die weit über die mathematischer Beweise hinaus geht. The übliche Strategie des

“Nachrechnens” or des schrittweisen Nachvollziehens verbietet sich dadurch. Anstelle einer vollständigen Überprüfung muss ein selektives Testen treten. Eine Teststrategie ist z.B. in [Sawitzki, 1994] beschrieben.

The Überprüfung is sowohl for die Implementierung als auch for den zu Grunde liegenden abstrakten Algorithmus nötig.

a:ch02:211

Aufgabe 2.18	
	for diese Aufgabenserie sei $y_i = a + bx_i + \varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0, \sigma^2)$ and $x_i = i, i = 1, \dots, 10$ .
	<p>Wählen Sie eine Strategie, um <code>lm()</code> im Hinblick auf den Parameterraum <math>(a, b, \sigma^2)</math> zu überprüfen.</p> <p>Gibt es eine naheliegende Zellzerlegung for die einzelnen parameter <math>a, b, \sigma^2</math>?</p> <p>Welche trivialen Fälle gibt es? Welche (uniforme) Asymptotik?</p> <p>Wählen Sie Testpunkte jeweils in der Mitte jeder Zelle and an den Rändern.</p> <p>Führen Sie diese Test durch and fassen Sie die Resultate zusammen.</p>
	<p>Welche Symmetrien/Antisymmetrien gibt es?</p> <p>Überprüfen Sie diese Symmetrien.</p>
	<p>Welche Invarianten/welches Covariate Verhalten gibt es?</p> <p>Überprüfen Sie diese Invarianten/Covariaten.</p>

a:ch02:221

Aufgabe 2.19	
	for diese Aufgabenserie sei $y_i = a + bx_i + \varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0, \sigma^2)$ .
	<p>Welche extremen Designs (<math>x_i</math>) gibt es? Überprüfen Sie das Verhalten von <code>lm()</code> bei vier extremalen Designs.</p>
	<p>Führen Sie die Aufgaben aus der letzten Gruppe aus, jetzt with variablem Design. Fassen Sie Ihren Bericht zusammen.</p>

a:ch02:23

Aufgabe 2.20	
	for diese Aufgabenserie sei $y_i = a + bx_i + \varepsilon_i$ with $\varepsilon_i$ iid $\sim N(0, \sigma^2)$ .
	<p>Modifizieren Sie <code>lm()</code> so, dass eine gesicherte function for das einfache lineare Modell entsteht, die auch Abweichungen von den Modellannahmen untersucht.</p>

#### 2.6.4 R Data Types

R is eine interpretierte Programmiersprache. Sie will es dem Anwender erlauben, Definitionen

`mode@mode|textit` and Konkretisierungen flexibel zu handhaben. Aus Geschwindigkeitsgründen versucht R, Auswertungen so spät wie möglich durchzuführen. Dies erfordert einige Einschränkungen an die Sprache, die R von anderen Programmiersprachen unterscheidet.

`attr@attr|textit` R kennt keine abstrakten Datentypen. Ein Datentyp ist durch seine Instanzen, die Variablen, die `class@class|textit` definiert.

`unclass@unclass|textit` Der Datentyp einer Variablen ist dynamisch: derselbe name in denselben Kontext kann zu unterschiedlichen Zeiten unterschiedliche Variablenvalues and Variablentypen kennzeichnen.

`print@print|textit` `printMethod` Zu jeder Variable einen bestimmten Typ. Das R-Typsysteem versteht

man jedoch am besten in seiner historischen Entwicklung und die entsprechenden Functions. In der ersten Stufe war der Typ beschrieben durch `mode()` (z.B. "numeric") und `storage.mode()` (z.B. "integer" or "real").

Beide Functions are weitgehend durch `typeof()` abgelöst. Eine Zusammenfassung der Typen, die durch `typeof()` derzeit berichtet werden, ist in [16] zu finden.

Komplexere Datentypen werden auf die in [16] definierten zurückgeführt, indem die Variablen with Attributen versehen werden. Dies geschieht with der function `attr()`, die auch benutzt werden kann, um Attribute zu setzen. So are eine Matrix or ein Array nur spezielle vectors, die sich dadurch auszeichnen, dass sie ein `dim`-Attribut haben. Das `class`-Attribut dient dazu, die Klasse explizit zu festzulegen.

for die wesentlichen Typen are Inspektionsprozeduren and Umwandlungsprozeduren: `is.(typ)()` prüft auf Typenzugehörigkeit, `as.(typ)()` wandelt den Typ.

### 2.6.5 Classes and Polymorphic Functions

ch02:polymorph

Im Zuge der Weiterentwicklung wurde eine Anleihe an objekt-orientierte Programmierung gemacht. Dafür wurde ein spezielles Attribut with dem names `class` genutzt: der name des Typs (or der "Klasse") wird hier gespeichert. Multiple Klassenzugehörigkeit in einer Hierarchie von Klassen is auch möglich. In diesem Fall enthält `class` einen vector von Klassennames. So hat zum Beispiel ein geordneter Faktor die Kennzeichnung `class = c("ordered", "factor")`. Zur Verwaltung der Klassen stehen Functions `class()`, `unclass()`, `inherits()` zur Verfügung.

The Klassenzuordnung basiert dabei auf Vertrauen. R überprüft nicht, ob die Datenstruktur with der angegebenen Klasse konsistent is.

Functions wie `plot()`, `print()` and viele weitere überprüfen die Typen- and Klassenzugehörigkeit ihrer Argument and verzweigen dann zu entsprechenden spezialisierten Functions. Dieses nennt man Polymorphismus. Wenn man eine polymorphe function auflistet, erhält man zunächst nur den Hinweise, das eine Dispatch-function `UseMethod()` aufgerufen wird. Beispiel:

expl:02-plot

*Example 2.12:*

<i>Input</i>	<i>Output</i>	UseMethod@UseMethod   textit polymorph methods@methods   textit lm@lm   textit lm@lm   textit
<code>plot</code>		
	<pre>function (x, y, ...) {   if (is.function(x) &amp;&amp; is.null(attr(x, "class")))     if (missing(y))       y &lt;- NULL     hasylab &lt;- function(...) !all(is.na(pmatch(names(list(...)),       "ylab")))     if (hasylab(...))       plot.function(x, y, ...)     else plot.function(x, y, ylab = paste(deparse(substitute(x)),       "(x)", ...))   }   else UseMethod("plot") } &lt;environment: namespace:graphics&gt;</pre>	

`UseMethod()` bestimmt die Klasse des ersten Argument, with dem die function aufgerufen wurde, sucht dann ein Spezialisierung for diese Klasse and ruft schliesslich die gefundene function auf. for **polymorphe** Functions findet man die entsprechenden Spezialisierungen with help von `methods()`, z.B. `methods(plot)`.

### 2.6.6 Extraktor Functions

Functions wie `lm()` liefern komplexe Datentypen with umfangreicher Information. In einer rein objekt-orientierten Umgebung würden Zugriffsmethoden with den Daten gemeinsam verkapselt. In R is Objekt-Orientierung in Ansätzen and auf verschiedene Weisen realisiert. Dies spiegelt zum Teil die Entwicklung wieder. Bei genügend verallgemeinerbaren Strukturen werden Zugriffsmethoden wie in Abschnitt 2.6.5 bereitgestellt. for die objects wie die von `lm()` gelieferten gibt es eine Reihe von Extraktor-Functions, die auf Komponenten zugreifen and diese geeignet aufbereiten.

<b>Extraktor-Funktionen for lm</b>	
<code>coef()</code>	extrahiert geschätzte Koeffizienten
<code>effects()</code>	extrahiert sukzessiv orthogonale Komponenten
<code>residuals()</code>	Roh-Residuen
<code>stdres()</code>	(in <code>library(MASS)</code> ) standardisierte Residuen

(Fortsetzung)→

<i>Extraktor-Funktionen for lm</i> (Fortsetzung)	
<i>studres()</i>	(in <i>library(MASS)</i> ) extern studentisiere Residuen
<i>fitted()</i>	
<i>vcov()</i>	Varianz/Kovarianzmatrix der geschätzten parameter
<i>predict()</i>	Konfidenz- und Toleranzintervalle
<i>confint()</i>	Konfidenz-Intervalle for parameter
<i>influence()</i>	extrahiert Einfluss-Diagnostiken
<i>model.matrix()</i>	bildet die Design-Matrix

## 2.7 Statistical Summary

Als Leitbeispiel diente in diesem Kapitel die statistische Analyse eines funktionellen Zusammenhangs. The betrachteten Modelle are finit in dem Sinne, dass ein endlich-dimensionaler Funktionenraum den in Betracht gezogenen Zusammenhang zwischen Regressoren and Respons beschreibt. The stochastische Komponente in diesen Modellen is noch auf eine (eindimensionale) Zufallsverteilung beschränkt. The Dimensionsbegriffe verdienen hier eine genauere Betrachtung. We haben zum einen die Regressor-Dimension. Dies is die Dimension des Raumes der beobachteten or abgeleiteten parameter. Nicht alle parameter are identifizierbar or schätzbar. Genauer gefasst is die Dimension die Vektorraum-Dimension des gewählten Modell-Raums. The Modelle werden durch parameter in diesem Raum beschrieben. Diese parameter können unbekannt or hypothetisch sein. In jedem Fall aber haben we sie als deterministisch betrachtet. Zum anderen haben we die stochastische Komponente, repräsentiert durch den Fehler-Term. In diesem Kapitel are we von homogenen Fehlern ausgegangen. Damit bestimmt der Fehler-Term im Prinzip eine Dimension, die allerdings aus a Raum von distributions stammt. for den Spezialfall der einfachen Gauß-linearen Modell are die distributions with zwei Parametern präzisiert, dem Erwartungswert and der Varianz. Von dem Erwartungswert haben we uns durch die Annahme befreit, dass das Modell im Mittel alle systematischen Effekte erfasst, also der Erwartungswert null is. The Varianz is in unseren Problemen noch ein unbekannter Störparameter. We haben die dadurch entstehenden Problemen vermieden, indem we uns auf Probleme beschränkt haben, in denen diese Störparameter durch einen geschätzten value ersetzt and so eliminiert wird.

## 2.8 Literature and Additional References

s:s02-bib

chh92sms

[3] Chambers, J.M.; Hastie, T.J. (eds.) (1992): Statistical Models in S. NewYork: Chapman & Hall.

jor93t1m

[11] Jørgensen, B. (1993): The Theory of Linear Models. NewYork: Chapman & Hall.

R:Rlang

[16] R Development Core Team (2004): The R language definition.

## LITERATURE AND ADDITIONAL REFERENCES

2-45

[21] Sawitzki, G. (1994): Numerical Reliability of Data Analysis Systems. Computational Statistics & Data Analysis 18.2 (1994) 269-286. <<http://www.statlab.uni-heidelberg.de/reports/>>.

[22] Sawitzki, G. (1994): Report on the Numerical Reliability of Data Analysis Systems. Computational Statistics & Data Analysis/SSN 18.2 (1994) 289-301. <<http://www.statlab.uni-heidelberg.de/reports/>>.

Generated by Sweave from:

```
Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S02reg.Rnw.tex,v  
Revision: 1.1  
Date: 2008/02/14 18:48:37  
name:  
Author: j40  
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S02reg.Rnw.tex,v $  
$Revision: 1.1 $  
$Date: 2008/02/14 18:48:37 $  
$name: $  
$Author: j40 $
```



---

## CHAPTER 3

---

# Comparison of Distributions

---

ch:03

We begin with der Konstruktion eines kleinen Werkzeugs, das uns Beispieldaten liefern wird. Basis ist ein kleiner Reaktionstester. We zeichen einen “zufälligen” Punkt, warten auf einen Maus-Klick, and registrieren die Position des Mauszeigers. Damit bei wiederholten Aufrufen das Bild stabil bleibt, fixieren we das Koordinatensystem.

ch03:Sframe03.1

*Example 3.1:*

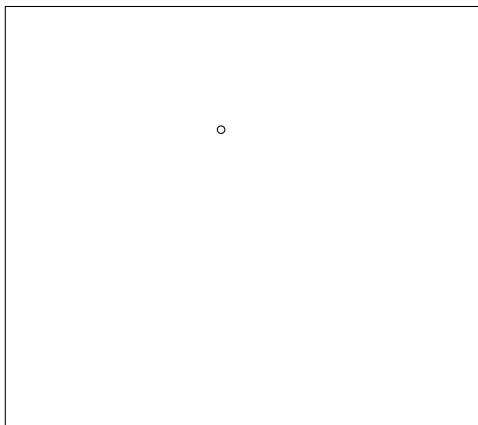
```
plot(x = runif(1), y = runif(1),
      xlim = c(0, 1), ylim = c(0, 1),
      main = "Bitte auf den Punkt klicken",
      xlab = '', ylab = '',
      axes = FALSE, frame.plot = TRUE)
locator(1)
```

Output

```
$x  
[1] 0.2817726
```

```
$y  
[1] 0.8596831
```

Bitte auf den Punkt klicken



We verpacken nun diesen Basistester. Wir merken uns die Koordinaten, versuchen, die Reaktionszeit des Benutzers zu messen, und liefern alle Resultate als Liste zurück.

ch03:Sframe03.2

**Example 3.2:**

```
click1 <- function() { Input
  x <- runif(1); y <- runif(1)
  plot(x = x, y = y, xlim = c(0, 1), ylim = c(0, 1),
       main = "Bitte auf den Punkt klicken",
       xlab = '', ylab = '',
       axes = FALSE, frame.plot = TRUE)
  clicktime <- system.time(xyclick <- locator(1))
  list(timestamp = Sys.time(),
       x = x, y = y,
       xclick = xyclick$x, yclick = xyclick$y,
       tclick = clicktime[3])
}
```

Zur weiteren Verarbeitung können wir die Liste in einen `data.frame` integrieren und diesen `data.frame` schrittweise with help von `rbind` erweitern.

**Example 3.3:**

```
dx <- as.data.frame(click1()) Input
dx <- rbind(dx, data.frame(click1()))
dx
```

	<i>Output</i>					
	timestamp	x	y	xclick	yclick	tclick
elapsed	2008-02-16 17:57:47	0.25553	0.14219	0.28177	0.86444	0.042
elapsed1	2008-02-16 17:57:48	0.69773	0.77130	0.68211	0.75986	0.883

a:03-click

Aufgabe 3.1	
	<p>Definieren Sie eine function <code>click(runs)</code>, die zu vorgegebener Anzahl <code>runs</code> die Aufgabe von <code>click1()</code> wiederholt und das Resultat als <code>data.frame</code> übergibt. Eine erste (zusätzliche) Messung sollte als Warmlaufen betrachtet werden und nicht in die Auswertung with einbezogen werden.</p> <p>Wählen Sie eine Anzahl <code>runs</code>. Begründen Sie Ihre Wahl von <code>runs</code>. Führen Sie <code>click(runs)</code> and speichern Sie das Resultat with help von <code>write.table()</code> in einer Datei.</p> <p>Stellen Sie die distribution der Komponente <code>tclick()</code> with den Methoden aus Kapitel <sup>ch01</sup>I (distribution function, Histogramm, Boxplot) dar.</p>

Shift-Familie | textbf

### 3.1 Shift/Scale Families

stochas-

tisch

kleiner | textbf

Ein Vergleich von distributions kann eine sehr anspruchsvolle Aufgabe sein. Der mathematische Raum, in dem distributions angesiedelt are, is nicht mehr ein numbersraum or ein (endlichdimensionaler) Vektorraum. Der eigentliche Raum, in dem distributions beheimatet are, is ein Raum von Maßen. In einfachen Fällen, etwa bei distributions auf  $\mathbb{R}$ , können we alles auf Verteilungsfunktionen reduzieren and are damit immerhin bei a Funktionenraum. Selbst hier kann ein Vergleich noch große Schwierigkeiten machen. We haben keine einfache Ordnungsrelation.

a:03-clickrl

Aufgabe 3.2	
	<p><a href="#">a:03-click</a></p> <p>Führen Sie Aufgabe 3.1 einmal with der rechten and einmal with der linken Hand durch. Vergleichen Sie die empirischen distributions von <code>tclick()</code>. The erhobenen Daten enthalten auch Information über die Positionen. Definieren Sie ein Maß <code>dist</code> for die Abweichung. Begründen Sie Ihre Definition. Führen Sie auch for <code>dist</code> einen rechts/links Vergleich durch.</p>

We konzentrieren uns hier auf den Vergleich von nur zwei distributions, etwa der von Messungen in zwei Behandlungsgruppen. Wie nehmen wieder einen einfachen Fall: die observations seien jeweils unabhängig identisch verteilt (jetzt with der for den Vergleich von Behandlungen üblichen Index-Notation).

$Y_{ij}$  unabhängig identisch verteilt with distribution function  $F_i$

$i = 1, 2$  Behandlungen

$j = 1, \dots, n_i$  observations in Behandlungsgruppe  $i$ .

Wie vergleichen we die observations in den Gruppen  $i = 1, 2$ ? The (einfachen) linearen Modelle

$$Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

betrachten als Unterschied häufig nur eine Verschiebung  $\Delta = \alpha_1 - \alpha_2$ .

**Bezeichnungen:** Zu einer distribution with distribution function  $F$  heißt die Familie with

$$F_a(x) = F(x - a)$$

die **Shift-Familie** zu  $F$ . The Verschiebung  $a$  heißt Shift- or Lage-parameter.

The Behandlung kann aber, in probabiliyen gesprochen, die probabiliysmassen auch in anderer Weise verschieben, als es ein additiver Term im Modell bewirken kann. We brauchen allgemeine Vergleichsmöglichkeiten als die durch einen Shift definierten.

**Bezeichnung:** Eine distribution with distribution function  $F_1$  heißt **stochastisch kleiner** als eine with distribution function  $F_2(F_1 \prec F_2)$ , wenn  $F_1$  eher bei kleineren valuesn liegt als  $F_2$ . Das bedeutet, dass  $F_1$  eher ansteigt.

$$F_1(x) \geq F_2(x) \forall x$$

and

$$F_1(x) > F_2(x) \text{ for mindestens ein } x.$$

for Shift-Familien gilt: Ist  $a > 0$ , so is  $F_a \prec F$ . Der Shift bewirkt eine Parallel-Verschiebung der Verteilungsfunktionen.

Ein typisches Resultat des Click-Experiments (Aufgabe 3.1 is zum Beispiel in Abbildung 3.1 [fig:03clickcdf](#))

dargestellt. The Zeiten for die rechte Seite are stochastisch kleiner als die for die linke Seite. The distributions gehören jedoch nicht zu einer Shift-Familie, denn die Verteilungsfunktionen are nicht parallel.

Skalen-  
Shiftfamilie|textbf

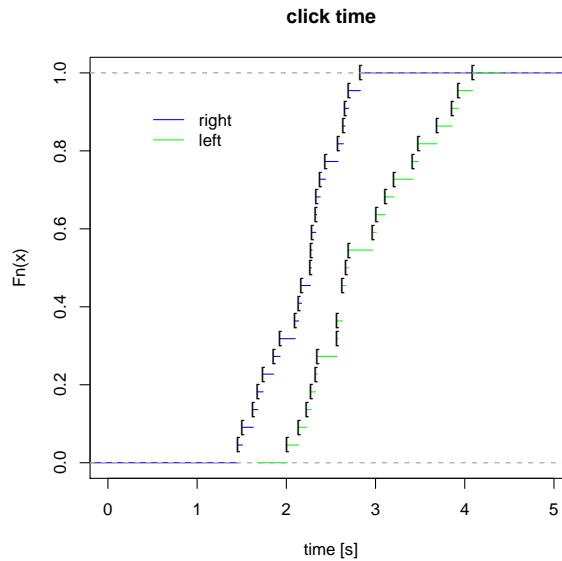


Figure 3.1 distribution function for die rechts/links-Klickzeit

fig:03clickcdf

a:ch03:1

Aufgabe 3.3	
	Wie sieht ein PP-Plot for $F_1$ gegen $F_2$ aus, wenn $F_1 \prec F_2$ ? Wie sieht ein QQ-Plot for $F_1$ gegen $F_2$ aus, wenn $F_1 \prec F_2$ ?

Leider is die dadurch definierte stochastische Ordnung nur von beschränktem value. Sie definiert keine vollständige Ordnung. for Shift-Familien is sie ausreichend. Aber Gegenexamples kann man sich konstruieren, wenn man die Shift-Familien nur geringfügig erweitert.

**Bezeichnungen:** Zu einer distribution with distribution function  $F$  heißt die Familie with

$$F_{a,b}(x) = F\left(\frac{x-a}{b}\right)$$

a:ch03:2 die *Skalen-Shiftfamilie* zu  $F$ .

qqnorm@qqnorm qqplot@qqplot	<b>Aufgabe 3.4</b>
	The Skalen-Shiftfamilien zur $N(0, 1)$ -distribution are die $N(\mu, \sigma^2)$ -distributions. Welche $N(\mu, \sigma^2)$ -distributions are stochastisch kleiner als die $N(0, 1)$ -distribution? Welche are stochastisch größer? for welche is die Ordnungsrelation undefiniert?

The aus der linearen Theorie kommende Einordnung nach Lage/Skalen and die stochastische Ordnung klaffen auseinander, and beide Aspekte müssen oft getrennt betrachtet werden. Viele statistische Methoden konzentrieren sich auf Aspekte, die durch Skalen-Shiftfamilien motiviert are. Unterschiede jenseits dessen, was durch Skala and Shift beschrieben werden kann, bedürfen oft besonderer Aufmerksamkeit.

In Kapitel 2 haben we eine typische Situation for lineare Modelle betrachtet. Im Prinzip haben we es with Skalen-Shiftfamilien zu tun. Der (stochastische) Skalenparameter in diesen Modellen is jedoch nur ein Störparameter, der eliminiert werden kann. Dazu benutzten we einen Schätzer for diesen Skalenparameter, die residuelle Varianz, die we dann heraus gekürzt haben. Als eine Besonderheit bei Gauß-linearen Modellen erhalten we hier unabhängige Schätzer for Erwartungswert and Varianz. Dadurch können we im Falle der einfachen Gauß-linearen Modelle Statistiken gewinnen, die nicht mehr vom Skalenparameter abhängen.

Im allgemeinen Fall haben we jedoch eine aufsteigende Leiter von Problemen:

- Shift-Alternativen
- Shift/Skalen-Alternativen
- stochastische Ordnung
- allgemeinere Alternativen

Test- and Schätzprobleme konzentrieren sich oft nur auf einen Aspekt des Problems, die Lage. Der Skalenparameter is hier nur eine Störgröße, ein “nuisance parameter”. Unterschiede im Shift-parameter führen zu stochastisch monotonen Beziehungen. Unterschiede im Skalenparameter are nicht so einfach einzuordnen and Test-Statistiken müssen erst von diesem, Störparameter bereinigt werden, wenn ausser dem Shift-parameter auch der Skalenparameter variieren kann.

### 3.2 QQ-Plot, PP-Plot, and Comparison of Distributions

s:3.2

Als Vergleichsdarstellung for Verteilungsfunktionen haben we den PP-Plot and den QQ-Plot kennengelernt. So lange man innerhalb einer Skalen-Shiftfamilie bleibt, hat der QQ-Plot zumindest in einer Hinsicht einen Vorteil gegenüber dem PP-Plot:

**Remark 3.1** Sind  $F_1, F_2$  Verteilungsfunktionen aus einer gemeinsamen Skalen-Shiftfamilie, so is der QQ-Plot von  $F_1$  gegen  $F_2$  eine Gerade.

Insbesondere for die Gaußverteilungen is der QQ-Plot gegen  $N(0, 1)$  ein wichtiges Hilfsmittel. Jede Gaußverteilung gibt in diesem Plot eine Gerade. Der QQ-Plot is for diese Situation bereits als function `qqnorm()` vorbereitet.

for den Vergleich von zwei Stichproben with gleichem Stichprobenumfang kann die entsprechende function `qqplot()` genutzt werden: bezeichnen we die empirischen Quantile with  $Y_{1,(i:n)}$

resp.  $Y_{2,(i:n)}$ , so ist dieser Plot der Graph  $(Y_{1,(i:n)}, Y_{2,(i:n)})_{i=1\dots n}$ . Sind die Stichprobenumfänge verschieden, so behilft sich R und generiert die Markierungspunkte durch lineare Interpolation, wobei der kleinere der beiden Stichprobenumfänge die Anzahl der Interpolationspunkte bestimmt.

Der *PP*-Plot hat keine dem *QQ*-Plot vergleichbare Äquivarianzeigenschaften. Wenn wir Skalen-Shiftparameter eliminieren wollen, müssen wir die Daten zunächst entsprechend transformieren. Die mathematische Theorie ist jedoch für den *PP*-Plot einfacher. Insbesondere gibt es auch hier einen entsprechenden Kolmogorov-Smirnov-Test (siehe Abschnitt 3.2.1).

Der Äquivarianz des *QQ*-Plots als Vorteil stehen auf der anderen Seite strukturelle Nachteile entgegen. In Bereichen niedriger Dichte bestimmen empirisch wenige Datenpunkte den Plot. Entsprechend hat er hier eine große Varianz. Gleichzeitig sind hier die probabilistisch benachbarte Quantile im Wertebereich weit entfernt: die große Varianz kombiniert sich ungünstig mit einer großen Variabilität, und der *QQ*-Plot zeigt entsprechend große Fluktuationen. Für die meisten Lehrbuch-distributions bedeutet dies, dass der *QQ*-Plot in den Randbereichen kaum zu interpretieren ist. Der *PP*-Plot hat keine entsprechenden Skalendefizite, aber auch nicht die Äquivarianzeigenschaft des *QQ*-Plots. Er wird deshalb in der Regel auf geeignete standardisierte Variable angewandt.

```
qqnorm@qqnorm|textbf
qqline@qqline
  (qqnorm)
qqplot@qqplot
  (qqnorm)
Topic
  hplot!qqnorm@qqnorm
Topic
  dis-
  tri-
  bu-
  tion!qqnorm@qqnorm
```

---

### help(qqplot)

---

<code>qqnorm</code>	<i>Quantile-Quantile Plots</i>
---------------------	--------------------------------

---

#### Description

`qqnorm` is a generic function the default method of which produces a normal QQ plot of the values in `y`. `qqline` adds a line to a normal quantile-quantile plot which passes through the first and third quartiles.

`qqplot` produces a QQ plot of two datasets.

Graphical parameters may be given as arguments to `qqnorm`, `qqplot` and `qqline`.

#### Usage

```
qqnorm(y, ...)
## Default S3 method:
qqnorm(y, ylim, main = "Normal Q-Q Plot",
      xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
      plot.it = TRUE, datax = FALSE, ...)

qqline(y, datax = FALSE, ...)

qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
       ylab = deparse(substitute(y)), ...)
```

`NA@NA|textit Arguments  
ppoints@ppoints|textit`

<code>x</code>	The first sample for <code>qqplot</code> .
<code>y</code>	The second or only data sample.
<code>xlab, ylab, main</code>	plot labels. The <code>xlab</code> and <code>ylab</code> refer to the y and x axes respectively if <code>datax = TRUE</code> .
<code>plot.it</code>	logical. Should the result be plotted?
<code>datax</code>	logical. Should data values be on the x-axis?
<code>ylim, ...</code>	graphical parameters.

#### *Value*

For `qqnorm` and `qqplot`, a list with components

<code>x</code>	The x coordinates of the points that were/would be plotted
<code>y</code>	The original y vector, i.e., the corresponding y coordinates <i>including NAs</i> .

#### *References*

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

#### *See Also*

`ppoints`, used by `qqnorm` to generate approximations to expected order statistics for a normal distribution.

#### *Examples*

```
y <- rt(200, df = 5)
qqnorm(y); qqline(y, col = 2)
qqplot(y, rt(300, df = 5))

qqnorm(precip, ylab = "Precipitation [in/yr] for 70 US cities")
```

a:03-clickqq

Aufgabe 3.5	
	Benutzen Sie den Quantil-Quantil-Plot, um die Resultate des rechts/links <code>click</code> -Experiments zu vergleichen. Formulieren Sie die Resultate.
	Fassen Sie die rechts/links <code>tclick</code> -Daten zu a vector zusammen. Vergleichen Sie den Quantil-Quantil-Plot with dem von Monte-Carlo-Stichproben aus dem zusammengefassten vector. Erinnerung: Zufallsstichproben können Sie with <code>sample()</code> ziehen. Mit <code>par(mfrow = c(2, 2))</code> teilen Sie den Zeichenbereich so ein, dass Sie vier Plots gleichzeitig sehen können.
	(Fortsetzung)→

Aufgabe 3.5	(Fortsetzung)
**	Benutzen Sie bei <code>sample()</code> den Parameterwert <code>replace = FALSE</code> . Wie müssen Sie jetzt <code>sample()</code> anwenden, um den zusammengefassten vector in zwei vectors with Monte-Carlo-Stichproben aufzuteilen? Welche Unterschiede zu <code>replace = TRUE</code> are zu erwarten?

a:ch03:3

Aufgabe 3.6	
	Bestimmen Sie for die <code>tclick</code> -Daten des rechts/links <code>click</code> -Experiments Skalen- und Shiftparameter so, dass die distributions in den Gruppen nach Skalen-Shift-Transformation möglichst gut übereinstimmen. Beschreiben Sie die Unterschiede anhand der Skalen-Shiftparameter. Verwenden Sie dazu eine Modellierung with a linearen Modell.
	Benutzen Sie die function <code>boxplot()</code> , um Quartile and Flankenverhalten darzustellen. Vergleichen Sie die Information with den Skalen-Shiftparametern. <i>Hinweis:</i> was entspricht dem Shift(Lage)parameter? Was entspricht dem Skalenparameter?

Wenn Darstellungen affin invariant are, können Skalen-Shiftparameter ignoriert werden. Wenn Darstellungen nicht affin invariant are, is es häufig hilfreich, zunächst Skalen-Shiftparameter geeignet zu schätzen, die distributions zu standardisieren, and dann die standardisierten distributions zu untersuchen.

Das Problem, das we uns damit potentiell einhandeln, is, dass dann das stochastiche Verhalten der Schätzung for die Skalen-Shiftparameter berücksichtigt werden muss. Der übliche Ausweg is es, vorsichtigerweise "konservative" Tests and robuste Schätzer zu benutzen. The folgende Transformation versucht, Skala and Lage an eine Standard-Normalverteilung anzupassen.

---

*Input*

```
ScaleShiftStd <- function (x) {
  xq <- quantile(x[!is.na(x)], c(0.25, 0.75))
  y <- qnorm(c(0.25, 0.75))
  slope <- diff(y)/diff(xq)
  (x-median(x, na.rm = FALSE)) * slope
}
```

---

Um distributions direkt miteinander vergleichen zu können, greifen we auf Techniken aus dem ersten Kapitel zurück. Was dort über den Vergleich zu einer theoretischen distribution gesagt worden is, kann analog auf den Vergleich von zwei distributions, z.B. aus zwei Behandlungsgruppen, übertragen werden. The statistischen Aussagen müssen jedoch revidiert werden. Nun betrachten we nicht mehr eine feste and eine zufällige distribution, sondern we vergleichen zwei zufällige (empirische) distributions.

The for den Einstichproben-Fall (eine Stichprobe im Vergleich zu einer hypothetischen distribution) benutzte Idee von Monte-Carlo-Bändern kann nicht unmittelbar übertragen werden: we wollen zwei distributions miteinander vergleichen, aber we haben keine ausgezeichnete Modelverteilung, aus der we Referenzstichproben ziehen können.

**Bootstrap|textbf{t}**

We können jedoch die Idee modifizieren und bedingte Monte-Carlo-Bänder konstruieren. Bedingt bedeutet hier: die Konstruktion hängt von beobachteten Stichprobenvaluesn ab. We nehmen an, dass we zwei Stichproben  $Y_{11}, \dots, Y_{1n_1}$  and  $Y_{21}, \dots, Y_{2n_2}$  von insgesamt unabhängigen and innerhalb der Gruppen identisch nach  $F_1$  resp.  $F_2$  verteilten observations haben. Falls kein Unterschied zwischen den distributions besteht, so is  $(Y_{11}, \dots, Y_{1n_1}, Y_{21}, \dots, Y_{2n_2})$  eine iid-Stichprobe aus einer gemeinsamen distribution  $F = F_1 = F_2$  with Stichprobenumfang  $n = n_1 + n_2$ . Bei einer iid-Stichprobe hätte jede Permutation der Indizes die gleiche probabiliy.

The motiviert das folgende Verfahren: we permutieren das Tupel  $(Y_{11}, \dots, Y_{1n_1}, Y_{21}, \dots, Y_{2n_2})$  and ordnen die ersten  $n_1$  values (nach Permutation) der ersten Gruppe zu, die anderen der zweiten.

The Permutationsgruppe is schnell so groß, dass sie nicht mehr vollständig ausgevaluest werden kann. Anstelle dessen benutzen we eine zufällige Auswahl von Permutationen. We benutzen die so generierten values, um Monte-Carlo-Bänder zu generieren.

**Ta:ch03:5**

ch03: stan-

dardize

scale

**ToDo:**

ch03: work

out

**ToDo:**

ch03: exer-

cise: power

Aufgabe 3.7	
	Modifizieren Sie die Functions for <i>PP-Plot</i> and <i>QQ-Plot</i> so, dass Monte-Carlo-Bänder for den Vergleich von zwei Stichproben hinzugefügt werden. <i>Hinweis:</i> with der function <code>sample()</code> können Sie zufällige Permutationen generieren.

Bei größerem Stichprobenumfang kann der Aufwand Permutationen zu generieren zu zeitaufwendig sein. Um Verwaltungsaufwand zu sparen, können we die Permutation durch ein Ziehen aus den  $n$  valuesn  $(Y_{11}, \dots, Y_{1n_1}, Y_{12}, \dots, Y_{1n_2})$  **with Zurücklegen** ersetzen. Diese approximative Lösung wird als **Bootstrap-Approximation**\* bezeichnet.

Da es nur endlich viele Permutationen gibt, können we bei kleinem Stichprobenumfang auch alle Permutationen durchgehen. We wählen die Bänder dann so, dass ein hinreichend großer Anteil (etwa mehr als 95 %) aller Kurven innerhalb der Bänder liegt. Permutationen, die sich nur innerhalb der Gruppen unterscheiden, ergeben dieselben Kurven. Diese Zusatzüberlegung zeigt, dass we nicht alle  $n!$  Permutationen überprüfen müssen, sondern nur die  $\binom{n}{n_1}$  Auswahlen for die Zuteilung zu den Gruppen.

**a:ch03:6**

Aufgabe 3.8	
**	Ergänzen Sie <i>PP-Plot</i> and <i>QQ-Plot</i> for die <i>click</i> -Experimente durch Permutations-Bänder, die 95 % der Permutationen abdecken.
*	Erzeugen Sie neue Plots, in denen Sie die <i>PP-Plots</i> and <i>QQ-Plots</i> durch Monte-Carlo-Bänder aus den Permutationen ergänzen. Benutzen Sie die Einhüllende von 19 Monte-Carlo-Stichproben. <i>Hinweis:</i> benutzen Sie die function <code>sample()</code> um eine Stichprobe vom Umfang $n_1$ aus $x = (Y_{11}, \dots, Y_{1n_1}, Y_{12}, \dots, Y_{1n_2})$ zu ziehen.
	(Fortsetzung)→

\* Vorsicht: es gibt beliebig wilde Definitionen von Bootstrap. Versuchen Sie stets, das Vorgehen mathematisch genau zu formulieren, wenn von Bootstrap die Rede is.

<b>Aufgabe 3.8</b>	(Fortsetzung)
	Erzeugen Sie neue Plots, in denen Sie die <i>PP</i> -Plots and <i>QQ</i> -Plots durch Monte-Carlo-Bänder aus den Permutationen ergänzen. Benutzen Sie die Einhüllende von 19 Bootstrap-Stichproben. <i>Hinweis:</i> Siehe <code>help(sample)</code> .

a:ch03:7

<b>Aufgabe 3.9</b>	
*	Versuchen Sie, die Eigenschaften der Permutationsbänder, Monte-Carlo-Bänder and Bootstrap-Bänder zu vergleichen, wenn $F_1 = F_2$ gilt.

Wenn nicht die distributions verglichen werden sollen, sondern nur einzelne festgelegte Kenngrößen, so können diese Strategien analog eingesetzt werden. Wenn we uns z.B. auf die Shift-Alternative beschränken (d.h.  $F_1$  and  $F_2$  are aus eine Shiftfamilie, d.h.  $F_1(x) = F_2(x - a)$  for ein  $a$ ), so können we etwa den Mittelwert (or den Median) als Kenngröße nehmen. Auf diese Kenngroße kann das obige Vorgehen analog angewandt werden, um zu entscheiden, ob die Hypothese, dass die distributions sich nicht unterscheiden ( $a = 0$ ), angesichts der Daten haltbar is.

a:ch03:8

<b>Aufgabe 3.10</b>	
*	Formulieren Sie die obigen Strategien for Intervalle for einzelne Teststatistiken (Beispiel: Mittelwert) anstelle for Bänder. <i>Hinweis:</i> Können Sie anstelle der zwei Mittelvalues for beide Gruppen eine eindimensionale zusammenfassende Statistik benutzen?

### 3.2.1 Kolmogorov Smirnov Tests

subs:3.4.2

In Kapitel 1 haben we den Kolmogorov-Smirnov-Test zum Vergleich einer Stichprobe  $(X_i)_{i=1,\dots,n}$  and der zugehörigen empirischen distribution  $F_n$  with einer (festen, vorgegebenen) distribution  $F$  kennengelernt. The kritische Testgröße is dabei

$$\sup |F_n - F|.$$

We können diesen Test etwas modifizieren, um zwei empirische distributions zu vergleichen. Anstelle der Modellverteilung  $F$  tritt nun eine zweite empirische distribution  $G_m$  von observations  $(Y_j)_{j=1,\dots,m}$  with zu Grunde liegender (unbekannter) distribution  $G$ . The kritische Testgröße is dann

$$\sup |F_n - G_m|.$$

Der darauf basierende Test is in der Literatur als 2-Stichproben-Kolmogorov-Smirnov-Test zu finden. Dieser Test korrespondiert zum *PP*-Plot and erlaubt es, Bänder zum *PP*-Plot zu konstruieren.

We können Bänder auch durch Simulation bestimmen. Im Gegensatz zum 1-Stichproben-Test haben we jetzt keine vorgegebene distribution, aus der we simulieren können. Unter der Hypothese, dass die distributions  $F$  and  $G$  sich nicht unterscheiden, verhält sich jedoch bei unabhän-

```
Test!t
t.test@t.test|textbf{t.est}
Topic
  ht-
est@t.test@t.est
  a:ch03:14
```

gigen observations der gemeinsame vector  $(X_1, \dots, X_n, Y_1, \dots, Y_m)$  wie ein vector von  $n + m$  unabhängigen random numbers with identischer distribution  $F = G$ . Bei gegebenen Daten kann diese Beziehung zur Simulation genutzt werden. Durch eine Permutation  $\pi$  der Indizes erzeugt **t.est** dem vector  $Z = (X_1, \dots, X_n, Y_1, \dots, Y_m)$  einen neuen vector  $Z'$  with  $Z'_i = Z_{\pi(i)}$ . The ersten  $n$  Komponenten benutzen we als simulierte values  $(X'_i)_{i=1,\dots,n}$ , die übrigen  $m$  Komponenten als simulierte values  $(Y'_j)_{j=1,\dots,m}$ .

Aufgabe 3.11	
*	Programmieren Sie diesen Algorithmus and ergänzen Sie den <i>PP</i> -Plot durch simulierte <i>PP</i> -Plots for eine kleine Anzahl (19?) von Permutation.
	Bestimmen Sie die Permutationsverteilung von $\sup  F_n - G_m $ aus den Simulation and berechnen Sie diesen value for die ursprünglichen Daten. Können Sie diesen Vergleich benutzen, um ein Testverfahren zu definieren?
	Der implementierte Kolmogorov-Smirnov-Test beinhaltet eine Approximation for den 2-Stichprobenfall. In unserer Simulation wissen we, dass we unter der Hypothese simulieren, die Hypothese also zutrifft. Untersuchen Sie die distribution des nominellen Niveaus unter den simulierten Bedingungen.

### 3.3 Tests for Shift Alternatives

s:3.3 Wenn we zusätzliche Verteilungsannahmen machen, können we andere Entscheidungsverfahren wählen. for diese Verfahren are aber die Verteilungsannahmen kritisch. Diese Abhängigkeit von den Verteilungsannahmen kann gemildert or vermieden werden, wenn we geeignete Verteilungsannahmen sicherstellen könnnen. Der *F*-Test, den we im letzten Kapitel kennengelernt haben, is ein Beispiel for ein verteilungsabhängiges Verfahren. for den Zwei-Stichprobenfall kann dieser Test modifiziert werden zum *t*-Test, der auch die Richtung des Unterschiedes widerspiegelt. (Das Quadrat der *t*-Statistik is eine *F*-Statistik.)

---

**help(t.test)**

---

<b>t.test</b>	<i>Student's t-Test</i>
---------------	-------------------------

---

*Description*

Performs one and two sample t-tests on vectors of data.

*Usage*

```
t.test(x, ...)
## Default S3 method:
t.test(x, y = NULL,
```

```
alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE,
conf.level = 0.95, ...)

## S3 method for class 'formula':
t.test(formula, data, subset, na.action, ...)
```

`model.frame@model.frame|t`

### Arguments

<code>x</code>	a numeric vector of data values.
<code>y</code>	an optional numeric vector data values.
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
<code>mu</code>	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
<code>paired</code>	a logical indicating whether you want a paired t-test.
<code>var.equal</code>	a logical variable indicating whether to treat the two variances as being equal. If <code>TRUE</code> then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
<code>conf.level</code>	confidence level of the interval.
<code>formula</code>	a formula of the form <code>lhs ~ rhs</code> where <code>lhs</code> is a numeric variable giving the data values and <code>rhs</code> a factor with two levels giving the corresponding groups.
<code>data</code>	an optional matrix or data frame (or similar: see <code>model.frame</code> ) containing the variables in the formula <code>formula</code> . By default the variables are taken from <code>environment(formula)</code> .
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
<code>...</code>	further arguments to be passed to or from methods.

### Details

The formula interface is only applicable for the 2-sample tests.

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are removed (in pairs if `paired` is `TRUE`). If `var.equal` is `TRUE` then the pooled estimate of the variance is used. By default, if `var.equal` is `FALSE` then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

```
prop.test@prop.test$`Value`
```

A list with class "htest" containing the following components:

<b>statistic</b>	the value of the t-statistic.
<b>parameter</b>	the degrees of freedom for the t-statistic.
<b>p.value</b>	the p-value for the test.
<b>conf.int</b>	a confidence interval for the mean appropriate to the specified alternative hypothesis.
<b>estimate</b>	the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.
<b>null.value</b>	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
<b>alternative</b>	a character string describing the alternative hypothesis.
<b>method</b>	a character string indicating what type of t-test was performed.
<b>data.name</b>	a character string giving the name(s) of the data.

#### See Also

`prop.test`

#### Examples

```
t.test(1:10,y=c(7:20))      # P = .00001855
t.test(1:10,y=c(7:20, 200)) # P = .1245    -- NOT significant anymore

## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)
## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t.test(extra ~ group, data = sleep)
```

In seiner einfachsten Form setzt der *t*-Test voraus, dass we unabhängig identisch verteilte Stichproben aus Normalverteilungen haben. Tatsächlich reichen schwächere Voraussetzungen. Wenn we die *t*-Test-Statistik als

$$t = \frac{\widehat{\mu}_1 - \widehat{\mu}_2}{\sqrt{Var(\widehat{\mu}_1 - \widehat{\mu}_2)}} \quad (3.1)$$

eq:03tstat

schreiben, so sehen we, dass *t* *t*-verteilt is, wenn  $\widehat{\mu}_1 - \widehat{\mu}_2$  normalverteilt and  $(Var(\widehat{\mu}_1 - \widehat{\mu}_2))$   $\chi^2$  verteilt is, and beide Term unabhängig are. Der zentrale Grenzwertsatz garantiert, dass  $\widehat{\mu}_1 - \widehat{\mu}_2$  unter milden Bedingungen zumindest asymptotisch normalverteilt is. Analoges gilt oft for  $(Var(\widehat{\mu}_1 - \widehat{\mu}_2))$ . Gilt die Unabhängigkeit beider Terme, so is *t* approximativ *t*-verteilt.

a03-tsim

Aufgabe 3.12	
*	<p>Bestimmen Sie in einer Simulation die distribution von <math>\bar{Y}</math>, <math>\widehat{Var}(Y)</math> und der <math>t</math>-Statistik for <math>Y</math> aus der uniformen distribution <math>U[0, 1]</math> with Stichprobenumfang <math>n = 1, \dots, 10</math>. Vergleichen Sie die distributions aus der Simulation with der entsprechenden Normal-, <math>\chi^2</math>- resp. <math>t</math>-distribution.</p> <p>Bestimmen Sie in einer Simulation die distribution von <math>\bar{Y}</math>, <math>\widehat{Var}(Y)</math> und der <math>t</math>-Statistik for <math>Y</math> aus einer Mischung, die zu 90% aus einer <math>N(0, 1)</math>- and zu 10% aus einer <math>N(0, 10)</math>-distribution stammt, with Stichprobenumfang <math>n = 1, \dots, 10</math>. Vergleichen Sie die distributions aus der Simulation with der entsprechenden Normal-, <math>\chi^2</math>- resp. <math>t</math>-distribution.</p>

Der  $t$ -Test hat eine gewisse Robustheit, die ihm eine approximative Gültigkeit geben kann. Man kann sich jedoch ganz von der Normalverteilungs-Voraussetzung befreien. Wenn we analog zum  $F$ -Test resp.  $t$ -Test vorgehen, aber anstelle der Urdaten die Ränge benutzen, gewinnen we Testverfahren, die verteilungsunabhängig are (zumindest, solange keine Bindungen auftreten können). Der Wilcoxon-Test ist eine verteilungsunabhängige Variante des  $t$ -Tests. Theoretisch entspricht er genau dem  $t$ -Test, angewandt auf die (gemeinsam) rangtransformierten Daten. Wie der  $t$ -Test is dieser Test nur darauf ausgelegt, die Nullhypothese (kein Unterschied) gegen eine Shift-Alternative zu testen. for die praktische Anwendung können arithmetische Vereinfachungen ausgenutzt werden. Deshalb is die Beziehung zwischen den üblichen Formeln for den  $t$ -Test and for den Wilcoxon-Test nicht einfach zu erkennen.

Um den Wilcoxon-Test anzuwenden, muss zum einen die Teststatistik berechnet werden. Zur Bestimmung kritischer values, with denen die Teststatistik zu vergleichen is, muss zum anderen die distribution function ausgevaluert werden. Sind alle observations paarweise verschieden, so hängt diese function nur von  $n_1$  and  $n_2$  ab, and relativ einfache Algorithmen stehen zur Verfügung. Diese are in der function R standardmäßig verfügbar and werden von `wilcox.test()` benutzt. Gibt es Bindungen in den Daten, d.h. gibt es übereinstimmende values, so hängt die distribution vom speziellen Muster dieser Bindungen ab and die Berechnung is aufwendiger. `wilcox.test()` greift in diesem Fall auf Approximationen zurück. Zur exakten (im Gegensatz zur approximativen) Auswertung stehen jedoch die entsprechenden Algorithmen ebenfalls zur Verfügung. Dazu benötigt man `library(coin)`. The exakte Variante des Wilcoxon-Tests findet sich dort etwa als `wilcox_test()`.

Auf den Rängen basierende verteilungsunabhängige Verfahren zu charakterisieren and with den früher vorgestellten verteilungsunabhängigen Monte-Carlo-Verfahren and deren Varianten zu vergleichen is ein klassischer Teil der Statistik. Literatur dazu findet man unter den Schlagworten "Rangtests" or "verteilungsfreie Verfahren". Zusätzliche R-Funktionen finden sich in `library(coin)` sowie in einigen speziellen Paketen.

Natürlich stellt sich die Frage nach dem Informationsverlust. Wenn we uns auf die Daten beschränken and keine or geringe Verteilungsannahmen machen, haben we weniger Information als in a Modell with expliziten Verteilungsannahmen. Wenn we die Daten auf die Ränge reduzieren, verschenken we zusätzlich möglicherweise Information. Dieser Informationsverlust kann z.B. durch die asymptotische relative Effizienz gemessen werden. Dies is (asymptotisch) der Stichprobenumfang eines optimalen Tests, der benötigt wird, eine vergleichbare Güte wie ein konkurrierender Test zu erreichen. Beim Wilcoxon-Test unter Normalverteilung hat dies einen value von 94%. Gilt also die Normalverteilungsannahme, so benötigt der (optimale)  $t$ -Test nur 94% des Stichprobenumfangs, die der Wilcoxon-Test benötigt. 6% des Stichprobenumfangs are die Kosten for die Reduzierung auf Ränge. Gilt die Normalverteilungsannahme nicht, so kann

est!Wilcoxon  
ilcox.test@wilcox.test|  
Bindung  
wilcox.test@wilcox.test|  
exakter\u2296Test  
Test!exakt  
wilcox\_test@wilcox\_test|

**ToDo:**  
add power  
function

**ToDo:**  
clarify: in-  
formation  
<- > ass-  
umptions

wilcox.test@wilcox.test | textbf{def} t-test möglicherweise zusammenbrechen. Der Wilcoxon-Test bleibt ein valider Test auf die Topic Shift-Alternative.  
**To Do:**  
add est power

---

simulation - see [help\(wilcox.test\)](#)

Rnw.tex  
source

---

DRAFT

Performs one and two sample Wilcoxon tests on vectors of data; the latter is also known as ‘Mann-Whitney’ test.

Usage

```
wilcox.test(x, ...)

## Default S3 method:
wilcox.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
            conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula':
wilcox.test(formula, data, subset, na.action, ...)
```

### *Arguments*

<code>x</code>	numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
<code>y</code>	an optional numeric vector of data values.
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of " <code>two.sided</code> " (default), " <code>greater</code> " or " <code>less</code> ". You can specify just the initial letter.
<code>mu</code>	a number specifying an optional location parameter.
<code>paired</code>	a logical indicating whether you want a paired test.
<code>exact</code>	a logical indicating whether an exact p-value should be computed.
<code>correct</code>	a logical indicating whether to apply continuity correction in the normal approximation for the p-value.
<code>conf.int</code>	a logical indicating whether a confidence interval should be computed.
<code>conf.level</code>	confidence level of the interval.
<code>formula</code>	a formula of the form <code>lhs ~ rhs</code> where <code>lhs</code> is a numeric variable giving the data values and <code>rhs</code> a factor with two levels giving the corresponding groups.

<code>data</code>	an optional matrix or data frame (or similar: see <code>model.frame</code> ) containing the variables in the formula <code>formula</code> . By default the variables are taken from <code>environment(formula)</code> .	<code>model.frame@model.frame@</code>
<code>subset</code>	an optional vector specifying a subset of observations to be used.	
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .	
<code>...</code>	further arguments to be passed to or from methods.	

### Details

The formula interface is only applicable for the 2-sample tests.

If only `x` is given, or if both `x` and `y` are given and `paired` is `TRUE`, a Wilcoxon signed rank test of the null that the distribution of `x` (in the one sample case) or of `x-y` (in the paired two sample case) is symmetric about `mu` is performed.

Otherwise, if both `x` and `y` are given and `paired` is `FALSE`, a Wilcoxon rank sum test (equivalent to the Mann-Whitney test: see the Note) is carried out. In this case, the null hypothesis is that the distributions of `x` and `y` differ by a location shift of `mu` and the alternative is that they differ by some other location shift.

By default (if `exact` is not specified), an exact p-value is computed if the samples contain less than 50 finite values and there are no ties. Otherwise, a normal approximation is used. Optionally (if argument `conf.int` is true), a nonparametric confidence interval and an estimator for the pseudomedian (one-sample case) or for the difference of the location parameters `x-y` is computed. (The pseudomedian of a distribution  $F$  is the median of the distribution of  $(u+v)/2$ , where  $u$  and  $v$  are independent, each with distribution  $F$ . If  $F$  is symmetric, then the pseudomedian and median coincide. See Hollander & Wolfe (1973), page 34.) If exact p-values are available, an exact confidence interval is obtained by the algorithm described in Bauer (1972), and the Hodges-Lehmann estimator is employed. Otherwise, the returned confidence interval and point estimate are based on normal approximations.

With small samples it may not be possible to achieve very high confidence interval coverages. If this happens a warning will be given and an interval with lower coverage will be substituted.

### Value

A list with class "htest" containing the following components:

<code>statistic</code>	the value of the test statistic with a name describing it.
<code>parameter</code>	the parameter(s) for the exact distribution of the test statistic.
<code>p.value</code>	the p-value for the test.
<code>null.value</code>	the location parameter <code>mu</code> .
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	the type of test applied.
<code>data.name</code>	a character string giving the names of the data.
<code>conf.int</code>	a confidence interval for the location parameter. (Only present if argument <code>conf.int = TRUE</code> .)
<code>estimate</code>	an estimate of the location parameter. (Only present if argument <code>conf.int = TRUE</code> .)

`psignrank@psignrank\textit{Warning}`  
`pwilcox@pwilcox\textit{Warning}`  
`wilcox.exact@wilcox\textit{exact}`  
This function can use large amounts of memory and stack (and even crash R if the stack limit is exceeded) if `exact = TRUE` and one sample is large (several thousands or more).  
`kruskal.test@kruskal\textit{test}`  
`t.test@t\textit{test}`

### *Note*

The literature is not unanimous about the definitions of the Wilcoxon rank sum and Mann-Whitney tests. The two most common definitions correspond to the sum of the ranks of the first sample with the minimum value subtracted or not: R subtracts and S-PLUS does not, giving a value which is larger by  $m(m + 1)/2$  for a first sample of size  $m$ . (It seems Wilcoxon's original paper used the unadjusted sum of the ranks but subsequent tables subtracted the minimum.)

R's value can also be computed as the number of all pairs  $(x[i], y[j])$  for which  $y[j]$  is not greater than  $x[i]$ , the most common definition of the Mann-Whitney test.

## *References*

Myles Hollander & Douglas A. Wolfe (1973), *Nonparametric statistical inference*. New York: John Wiley & Sons. Pages 27–33 (one-sample), 68–75 (two-sample).  
Or second edition (1999).

David F. Bauer (1972), Constructing confidence sets using rank statistics. *Journal of the American Statistical Association*, **67**, 687–690.

#### *See Also*

`psignrank`, `pwilcox`.

`wilcox.exact` in `exactRankTests` covers much of the same ground, but also produces exact p-values in the presence of ties.

`kruskal.test` for testing homogeneity in location parameters in the case of two or more samples; `t.test` for an alternative under normality assumptions [or large samples]

### *Examples*

```

## Two-sample test.
## Hollander & Wolfe (1973), 69f.
## Permeability constants of the human chorioamnion (a placental
## membrane) at term (x) and between 12 to 26 weeks gestational
## age (y). The alternative of interest is greater permeability
## of the human chorioamnion for the term pregnancy.
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
wilcox.test(x, y, alternative = "g")      # greater
wilcox.test(x, y, alternative = "greater",
            exact = FALSE, correct = FALSE) # H&W large sample
                                         # approximation

wilcox.test(rnorm(10), rnorm(10, 2), conf.int = TRUE)

## Formula interface.
boxplot(Ozone ~ Month, data = airquality)
wilcox.test(Ozone ~ Month, data = airquality,
            subset = Month %in% c(5, 8))

```

a:ch03:11

Aufgabe 3.13	
	Benutzen Sie den Wilcoxon-Test, um die Resultate des rechts/links <i>click</i> -Experiments zu vergleichen.

a:ch03:12

Aufgabe 3.14	
***	<p>Beim rechts/links <i>click</i>-Experiment are mehrere Effekte vermischt. Einige Probleme:</p> <ul style="list-style-type: none"> <li>• The Antwortzeit beinhaltet Reaktionszeit, Zeit for die Grob-Bewegung der Maus, Zeit for die Fein-Adjustierung etc.</li> <li>• for rechts-links-Bewegungen reicht in der Regel ein Schwenken der Hand aus. for vorwärts-rückwärts-Bewegungen is in der Regel eine Arm-Bewegung nötig. Es is nicht zu erwarten, dass beide vergleichbares statistisches Verhalten haben.</li> <li>• Bei aufeinanderfolgenden Registrierungen kann es zum einen Trainings- zum anderen Ermüdungseffekte geben.</li> </ul> <p>Können Sie Experiment and Auswertung so modifizieren, dass Unterschiede in der Reaktionszeit untersucht werden können?</p> <p>Können Sie Experiment and Auswertung so modifizieren, dass Unterschiede in der Genauigkeit der Endposition untersucht werden können?</p>
***	<p>Untersuchen and dokumentieren Sie for sich rechts-links-Unterschiede in der Reaktionszeit and in der Genauigkeit. Formulieren Sie ihr Resultat als Bericht.</p>

a:ch03:13

Aufgabe 3.15	
	<p>Betrachten Sie als Verteilungsfamilien die Shift/Skalenfamilien von <math>N(0, 1)</math> and <math>t(3)</math>. Entwerfen Sie ein Szenario, um den Wilcoxon-Test with dem <math>t</math>-Test jeweils innerhalb dieser Familien zu vergleichen.</p> <p>Führen Sie diesen Test in einer Simulation for Stichprobenumfänge <math>n_1 = n_2 = 10, 20, 50, 100</math> durch and fassen Sie die Resultate zusammen.</p> <p>Führen Sie eine analoge Simulation for die Lognormal-Verteilungen durch.</p>

### 3.4 Power and Confidence

**ToDo:** Simulation, Power

#### 3.4.1 Theoretical Power and Confidence

Am Beispiel des t-Tests können we illustrieren, wie ein Test aufgebaut is. Der Test benutzt eine Teststatistik, hier die  $t$ -Test-Statistik zum Vergleich zweier Gruppen (3.1). We kennen die distribution dieser Statistik: for den  $t$ -Test is bei unabhängigen normalverteilten Fehlern and gleicher Varianz die Teststatistik  $t(n_1 + n_2 - 2)$  verteilt. Zu gewähltem Niveau  $\alpha$  können we aus der distribution function können we Grenzen ablesen, die bei dieser distribution nur with einer probabiliy  $\alpha$  unter- resp. überschritten werden. Benutzen we beide Grenzen, so erhalten we einen zweisietigen Bereich with der Irrtumsprobabiliy  $2\alpha$ .

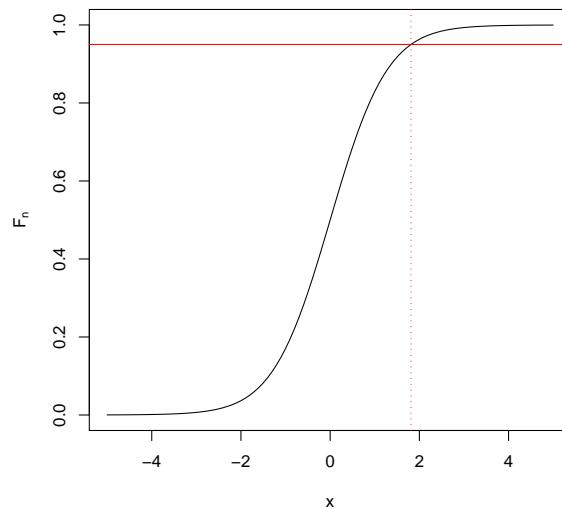
fig:ch03testcrit

**Example 3.4:***Input*

```

n1<- 6; n2 <- 6
df <- n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]))
abline(h=1-alpha, col="red")      # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value

```



Wollen we z.B. die Hypothese  $\mu_1 = \mu_2$  gegen die Alternative  $\mu_1 > \mu_2$  testen, so wählen we als Verwerfungsbereich den Bereich über der oberen dieser Grenze. We wissen, dass we bei Gültigkeit der Hypothese höchstens with probabiliy  $\alpha$  zufällig eine Testgröße in diesem Bereich bekommen.

for den  $t$ -Test wissen we sogar mehr. Unter den Modellvoraussetzungen unabhängig normalverteilter Fehler and gleicher Varianz is die  $t$ -Test-Statistik immer  $t$  verteilt. Auf der Hypothese is sie  $t$  verteilt with Nichzentralitätsparameter 0, folgt also der zentralen  $t$ -distribution. Auf der Alternative haben we eine  $t$ -distribution with Nichzentralitätsparameter  $(\mu_1 - \mu_2)\sigma^{-1}\sqrt{n_1 n_2 / (n_1 + n_2)}$ . Damit kann for jede Alternative unter den Modellannahmen die Stärke des Tests abgelesen werden, d.h. die probabiliy bei Vorliegen dieser Alternative die Hypothese zu verwerfen.

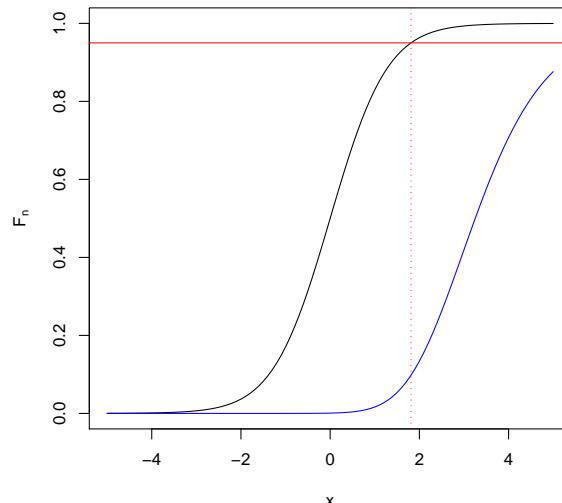
**To Do:**  
introduce  
noncentral  
 $t$  in ch. 2

ig:ch03testcritis

*Example 3.5:*

*Input*

```
n1<- 6; n2 <- 6
df <- n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]))
abline(h=1-alpha, col="red")      # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value
n1 <- 5
n2 <- 5
n <- n1+n2
theta <- 2
ncp <- theta * sqrt(n1 * n2/(n1+n2))
curve(pt(x,df=df, ncp=ncp),add=TRUE, col="blue")
```



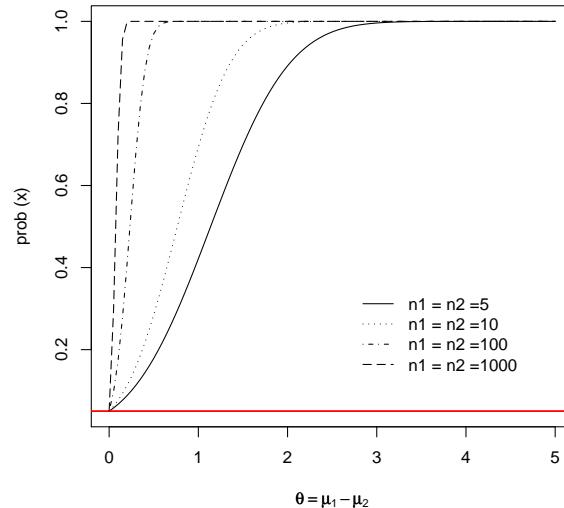
The Güte des Tests können wir darstellen, indem wir die Verwerfungsprobabilität in Abhängigkeit von  $(\mu_1 - \mu_2)\sigma^{-1}$  auftragen.<sup>†</sup>

fig:ch03testpow

<sup>†</sup> Konventionell wird er die Gütfunktion nur auf der Alternative, d.h z.B. für  $(\mu_1 - \mu_2) > 0$  betrachtet. We setzen sie hier auch auf der Hypothese, d. h. für  $(\mu_1 - \mu_2) > 0$  fort.

**Example 3.6:**

```
Input
tpower <- function(n1, n2, alpha,...){
  df <- n1 + n2 -2
  tlim <- qt(1-alpha,df=df)
  prob <- function(theta){
    pt(tlim, df = df,
       ncp = theta * sqrt(n1 * n2/(n1+n2)),
       lower.tail=FALSE)
  }
  curve(prob, 0, 5, xlab=expression(theta==mu[1]-mu[2]), ...)
  abline(h=alpha, col="red")
}
tpower(5, 5, 0.05)
tpower(10, 10, 0.05, add =TRUE, lty = 3)
tpower(100,100, 0.05, add =TRUE, lty = 4)
tpower(1000, 1000, 0.05, add =TRUE, lty = 5)
legend("bottomright",
       lty=c(1,3,4,5),
       legend=c("n1 = n2 =5", "n1 = n2 =10",
               "n1 = n2 =100", "n1 = n2 =1000"),
       inset=0.1, bty="n")
```



Der hier benutzte Zusammenhang kann auch benutzt werden, um zu bestimmen, wie groß der Stichprobenumfang sein muss, um auf der Hypothese höchstens mit einer Wahrscheinlichkeit  $\alpha$  fälschlich zu verwerfen, bei Vorliegen einer spezifizierten Alternative jedoch mit einer gewählten Wahrscheinlichkeit die Hypothese richtigerweise zu verwerfen. Dazu gibt es die vorbereitete Funktion `power.t.test()`.

ch03powert

power.t.test@power.t.test

**Example 3.7:**

```
power.t.test(delta=2,
             power=0.8,
             sig.level=0.01,
             type="two.sample",
             alternative="one.sided")
```

*Input*

```
Two-sample t test power calculation
```

```
n = 6.553292
delta = 2
sd = 1
sig.level = 0.01
power = 0.8
alternative = one.sided
```

*Output*

NOTE: n is number in \*each\* group

### 3.4.2 Simulated Power and Confidence

Sind die theoretischen Eigenschaften einer Test-Statistik bekannt, so ist dies der beste Weg, die Güte zu analysieren. In einer Umgebung wie R haben wir die Möglichkeit, die Güte auch dann zu untersuchen, wenn theoretische Resultate nicht vorliegen oder nicht zugänglich sind. Zu festgelegten Alternativen können wir Zufallsstichproben generieren, Tests durchführen und den relativen Anteil der Verwerfungen bestimmen. Generieren wir *nsimul* unabhängige Zufallsstichproben mit identischer Verteilung, so ist die Anzahl der Verwerfungen binomialverteilt und

$$\hat{p} = \frac{\# \text{Verwerfungen}}{nsimul}$$

ein Schätzer für die Verwerfungsprobabilität.

Als Beispiel untersuchen wir, wie sich der *t*-Test verhält, wenn die Daten lognormal verteilt sind. Wir vergleichen zwei Gruppen jeweils mit Stichprobenumfang  $n_1 = n_2 = 10$ , zunächst auf der Hypothese:

ch03tlognH

*Example 3.8:*

```
prop.test@prop.test|text{  
nsimul <- 300  
n1<- 10; n2 <- 10  
alpha <- 0.01  
x <- 0  
for (i in 1:nsimul) {  
  if (t.test(exp(rnorm(n1)),exp(rnorm(n2))),  
      alternative="less",  
      var.equal = TRUE)$p.value < alpha){  
    x <- x+1}  
  }  
p <- x/nsimul  
cat("estim p", p)
```

Input

estim p 0.003333333

Output

The function `prop.test()` berechnet nicht nur diesen Schätzer, sondern auch einen Konfidenzbereich.

*Example 3.9:*

```
prop.test(n=nsimul, x=x)
```

Input

```
prop.test@prop.test|text{  
1-sample proportions test with continuity correction  
  
data: x out of nsimul, null probability 0.5  
X-squared = 294.03, df = 1, p-value < 2.2e-16  
alternative hypothesis: true p is not equal to 0.5  
95 percent confidence interval:  
 0.0001740250 0.0213609800  
sample estimates:  
      p  
0.003333333
```

Output

Analog z. B. wenn für die Alternative  $\log(x_2)$  nach  $N(\mu_2, 1)$  mit  $\mu_2 = 1$  verteilt ist:

```

power.prop.test@power.prop.test|textit
Variationskoeffizienten:
Example 3.10: Input
  nsimul <- 300
  n1<- 10; n2 <-10
  alpha <- 0.01
  x<-0
  for (i in 1:nsimul) {
    if (t.test(exp(rnorm(n1)),exp(rnorm(n2, mean = 1)),
               alternative="less",
               var.equal = TRUE)$p.value < alpha){
      x <- x+1}
  }
  p <- x/nsimul
  cat("estim p", p)

Output
estim p 0.1933333

prop.test(n = nsimul, x = x) Input
Output
  1-sample proportions test with continuity correction

data: x out of nsimul, null probability 0.5
X-squared = 111.63, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
  0.1511354 0.2435719
sample estimates:
  p
0.1933333

```

In `library(binom)` finden sich eine Reihe von Werkzeugen for eine differenziertere Analyse der Binomialverteilung.

The Konfidenzintervalle in diesem Beispiel zeige, dass hier ein Simulationsumfang von  $nsimul = 300$  nur grobe Ergebnisse liefert. for die Simulation wollen we die Genauigkeit besser kontrollieren. Den Simulationsumfang können we so wählen, dass eine gewünschte Genauigkeit erreicht werden kann. Eine genaue Planung können we with `power.prop.test()` machen. for Simulationszwecke reicht oft schon eine Abschätzung.

Mit  $\hat{p} := Z/n$  als Schätzer einer probabiliy  $p$  haben we  $E(\hat{p}) = p$  and  $Var(\hat{p}) = p(1 - p)/n$ . Ist  $p$  tatsächlich der interessierende parameter, so are Fehler relativ zum Zielparameter zu messen. Bei  $p = 50\%$  is ein Fehler von  $\pm 1\%$  anders zu bevaluesn, als bei  $p = 99\%$ . Der relative Fehler, der **Variationskoeffizient**, is

$$\frac{\sqrt{Var(\hat{p})}}{E(\hat{p})} = \sqrt{\frac{1-p}{np}}.$$

Um einen Variationskoeffizienten von höchsten  $\eta$  zu erhalten, brauchen we eine Stichproben-

umfang

$$n \geq \frac{1-p}{p\eta^2}.$$

Sind  $n$  and  $p$  in einer Größenordnung, bei der eine Normalapproximation gilt, so haben we for ein Konfidenzniveau  $1 - \alpha$  ein approximatives Konfidenzintervall with Grenzen

$$\hat{p} \pm \phi_{1-\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}.$$

Soll das Konfidenzintervall eine Länge von  $\eta p$  nicht überschreiten, so brauchen we einen Stichprobenumfang

$$n \geq \frac{\phi_{1-\alpha/2}^2(1-p)}{p\eta^2}.$$

Wie üblich is eine Wahl von  $\alpha$  zu treffen. for z. B.  $\alpha = 1\%$  with  $\phi_{1-\alpha/2} = 2.575829$  erhalten we die values in Tabelle 3.26. Falls we with höheren Quantilen arbeiten, werden we versuchen, die Fehler relativ zu  $1 - p$  zu beschränken. examples are in Tabelle 3.26.

**To Do:** adjust gray formats in tables

Table 3.26 Erforderlicher Stichprobenumfang for zweiseitige Konfidenzintervalle with relativer Länge  $\leq \eta$

$p$	$1 - p$	$n(\alpha = 10\%)$	$\eta = 0.01$	$n(\alpha = 1\%)$	$\eta = 0.01$
		$\eta = 0.1$		$\eta = 0.1$	
0.500	0.500	271	27 055	663	66 349
0.250	0.750	812	81 166	1990	199 047
0.100	0.900	2435	243 499	5971	597 141
0.010	0.990	26 785	2 678 488	65 685	6 568 547
0.001	0.999	270 283	27 028 379	662 826	662 822 617

tab:simulneta

Zu merken are die groben numbers: um with 90% Konfidenz eine probabiliy im Bereich von  $50\% \pm 5\%$  zu schätzen, are ca. 300 Simulationen notwendig. Um einen value bei 99% bis auf  $\pm 0.1\%$  genau zu schätzen are 30000 Simulationen nötig.

### 3.4.3 Quantilschätzung durch Simulation

The andere Seite des Problems is es, ein Quantil anhand einer Stichprobe zu schätzen. We wissen bereits, dass for eine Zufallsvariable  $X$  with stetiger distribution function  $F$  die Variable  $F(X)$  eine uniforme distribution auf  $[0, 1]$  hat. for die Quantilschätzung benötigen we die distribution function, ausgevaluest an den aus Ordnungsstatistiken. Diese haben we bereits in Kapitel ?? kennen gelernt. Dort hatten we als Theorem ??.

thm:simulF

**Theorem 3.2** Sind  $X_i, i = 1, \dots, n$  unabhängige observations aus einer distribution with stetiger distribution function  $F$  and is  $X_{(k:n)}$  die  $k$ . Ordnungsstatistik daraus, so is

$$F(X_{(k:n)}) \sim P_{beta}(\cdot; k, n - k + 1).$$

We wiederholen:

**Remark 3.3** Im allgemeinen ist die beta-distribution schief. Der Erwartungswert der Beta( $k, n-k+1$ )-Verteilung ist  $k/(n+1)$ . Um eine unverzerrte Schätzung des Quantils  $x_p$  zu erhalten, benutzt man  $X_{(k:n)}$  mit  $k/(n+1) = p$ . Die ‘‘plug in’’-Approximation  $k/n = p$  gibt eine verzerrte Schätzung.

Das Theorem kann direkt angewendet werden, um eine obere or untere Abschätzung for Quantile zu gewinnen. Insbesondere können we versuchen, das Minimum der beobachteten valuess  $X_{(1:n)}$  als untere Abschätzung for das  $p$ -Quantil. zu benutzen Das Konfidenzniveau is

$$P(X_{(1)} \leq F_p) = P(F(X_{(k)}) \leq p) = I_p(1, n),$$

wobei  $I$  das unvollständige Beta-Integral ist. für die speziellen Parameter  $(1, n)$  vereinfacht sich die Beta-Dichte zu  $n(1-p)^{n-1}$  und wir bekommen für das unvollständige Beta-Integral  $I_p(1, n) = 1 - (1-p)^n$ . Daraus folgt

$$P(X_{(1)} \leq F_p) = 1 - (1-p)^n$$

and we können ein Konfidenzniveau  $1 - \alpha$  sicherstellen, wenn

$$n \geq \frac{\ln \alpha}{\ln(1-p)}.$$

Der beobachtete Höchstwert kann als obere Abschätzung für das  $p$ -Quantil verwendet werden. Aus Symmetriegründen erhalten wir einen Konfidenzniveau von  $1 - \alpha$ , wenn

$$n \geq \frac{\ln \alpha}{\ln p}.$$

examples are in Tabelle 3.27 angegeben.

Table 3.27 Benötigter Stichprobenumfang zur Abschätzung eines Quantils mit Konfidenzniveau  $\geq 1 - \alpha$

p		n			
$X_{(1)} \leq F_p$	$X_{(n)} \geq F_p$	$\alpha = 10\%$	$\alpha = 5\%$	$\alpha = 1\%$	$\alpha = 0.5\%$
0.500	0.500	4	5	7	8
0.250	0.750	9	11	17	19
0.100	0.900	22	29	44	51
0.010	0.990	230	299	459	528
0.001	0.999	2302	2995	4603	5296

Zu merken are wieder die groben numbers: um eine einseitige Abschätzung für ein 1% (99%-Quantil einer stetigen distribution function with einer Konfidenz von 99% zu erhalten, werden beinahe 500 Simulationen benötigt.

We können einseitige Schranken zu Intervallen verknüpfen. Das entsprechende Resultat zur Berechnung der probabilty von Intervallen ist in Korollar [77](#):

---

cor:sumul2s

**Corollary 3.4** Mit der  $k_1$ -ten and  $k_1+k_2$ -ten Ordnungsstatistik ist das Intervall  $(X_{(k_1:n)}, X_{(k_1+k_2:n)})$  ein Konfidenzintervall für das  $p$ -Quantil mit der Überdeckungsprobabilität

$$I_n(k_1, n - k_1 + 1) - I_n(k_1 + k_2, n - k_1 - k_2 + 1).$$

The Simulationsumfänge zur Abschätzung von Quantilen are drastisch geringer als diejenigen, die zur vergleichbaren Schätzung von probabiliyen benötigt werden. Im Nachhinein is dies nicht verwunderlich: die Frage, ob eine Beobachtung über a bestimmten Quantil liegt, is einfacher, als die Aufgabe, den  $p$ -value zu schätzen. In Abschnitt sec:simul-mct werden we sehen, dass der notwendige Stichprobenumfang noch einmal drastisch verringert werden kann, wenn die Fragestellung auf ein Testproblem reduziert wird.

Ohne weitere Verteilungsannahmen gibt dies eine erste Möglichkeit, den Umfang einer Simulation festzulegen. In speziellen Situationen können geschickte Einfälle eine bedeutende Reduzierung des Stichprobenumfangs erlauben. Zunächst aber are die obigen Abschätzungen die Grundlage for Simulationen.

### 3.5 Qualitative Features of Distributions

**ToDo:**  
Qualitative  
features



**3.6 R Complements**

s:3.4



### 3.7 Statistical Summary

Als Leitbeispiel diente in diesem Kapitel der Vergleich von Stichproben. In einfachen Fällen unterscheiden sich Stichproben nur um eine Verschiebung des Mittelwerts. In diesem Fall können die Probleme auf die Ansätze aus Kapitel 2 reduziert werden. In diesem reduzierten Fall stimmen die um den Mittelwert zentrierten distributions überein. for den allgemeineren Fall, den we jetzt untersucht haben, gilt diese Vereinfachung nicht. Ein wichtiges Beispiel ist etwa die Untersuchung von Therapie-Studien. Hat eine Behandlung einen homogenen Effekt, so können we diesen with den Mitteln von Kapitel 2 untersuchen. Häufig aber gibt es unter einer Behandlung eine Aufspaltung in "Responder" und "Nicht-Responder". Dies geht über die in Kapitel 2 skizzierten Modelle hinaus, and die allgemeineren Ansätze aus diesem Kapitel 3 werden nötig.

We haben uns hier auf den Vergleich von zwei Stichproben beschränkt. The Praxis führt oft auf andere Probleme. So is ein typischer Fall, dass eine neue Behandlung with einer bekannten Referenz-Behandlung verglichen werden soll, wobei for die neue Behandlung nur eine Stichprobeninformation, for die Referenz-Behandlung aber umfassendere Vorinformation bereit steht. or eine Referenz-Behandlung soll with einer Serie von Alternativ-Behandlungen verglichen werden. Diese Probleme gehen über den Rahmen unserer Einführung hinaus. Hier kann nur auf weiterführende Literatur, z.B [T2] verwiesen werden.

### 3.8 Literature and Additional References

- [vr02mass] Venables, W.N.; Ripley, B.D. (2002): Modern Applied Statistics with S. Heidelberg: Springer
- [vr00pis] Venables, W.N.; Ripley, B.D. (2000): S Programming. Heidelberg: Springer
- [mil8issi] Miller, R. G. (1981): Simultaneous Statistical Inference. Heidelberg: Springer

```
Generated by Sweave from:
Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S03vergl.Rnw.tex,v
Revision: 1.1
Date: 2008/02/14 18:48:38
name:
Author: j40
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S03vergl.Rnw.tex,v $
$Revision: 1.1 $
$Date: 2008/02/14 18:48:38 $
$name: $
$Author: j40 $
```



## CHAPTER 4

# Dimensions 1, 2, 3, ..., $\infty$

---

ch:04

### 4.1 R Complements

In diesem Kapitel beginnen wir mit Ergänzungen zu R, um uns dann auf statistische Fragen zu konzentrieren. Wir werfen einen Blick auf die graphischen Möglichkeiten, die uns zur Verfügung stehen. Die Basis-Graphik von R ist an ein Plotter-Modell orientiert. Die Graphik folgt den Möglichkeiten, die das Zeichnen mit einem Stift bietet. Neben den ein- und zweidimensionalen Möglichkeiten, die wir bis jetzt kennengelernt haben, gibt es Möglichkeiten, eine Funktion darzustellen, die über ein Raster definiert ist. Dazu stehen im wesentlichen drei Functions zur Verfügung.

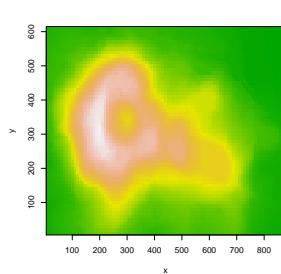
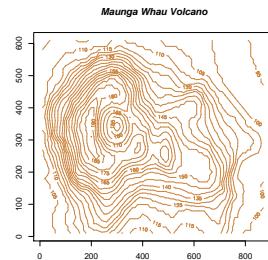
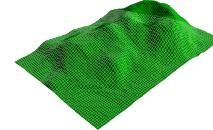
<b>3d-Graphik</b>	
<code>image()</code>	Gibt die Werte einer Variablen <code>z</code> in Graustufen oder Farbcodierung wieder.
<code>contour()</code>	Gibt die Konturen einer Variablen <code>z</code> .
<code>persp()</code>	Gibt einen perspektivischen Plot einer Variablen <code>z</code> .

ch04-SframeVolcano `image()` und `contour()` können auch benutzt werden, um andere Plots zu überlagern.

```
print@print|textit
Example 4.1:
```

*Input*

```
#oldpar <- par(mfrow=c(1,3))
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)
contour(x, y, volcano, levels = seq(90, 200, by = 5),
        col = "peru", main = "Maunga Whau Volcano", font.main = 4)
z <- 2 * volcano      # Exaggerate the relief
x <- 10 * (1:nrow(z)) # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z)) # 10 meter spacing (E to W)
## Don't draw the grid lines : border = NA
#par(bg = "slategray")
persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75, border = NA, box = FALSE)
```

*image()**contour()**persp()*

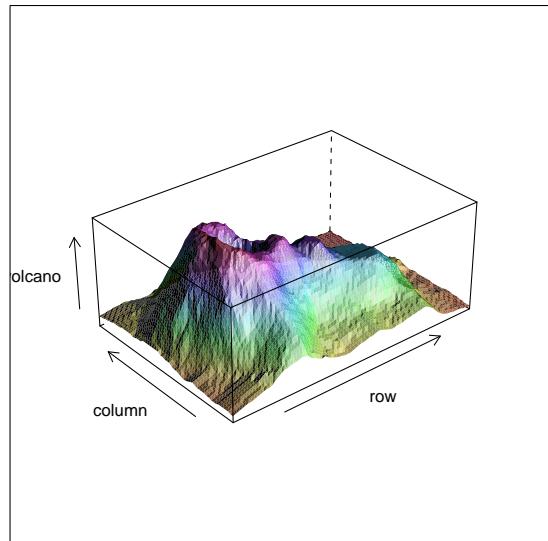
The Basis-Graphik is einfach zu handhaben, aber limitiert. Ein neues Grafiksystem arbeitet konzeptuell with Objekten and a Kamera-Modell. The Grafik-objects können kombiniert and bearbeitet werden. The Darstellung erfolgt in a getrennten Schritt. Einfache 2d-Grafiken können hier nachbearbeitet werden. for eine 3d-Darstellung können wie bei einer Kamera Abstand, Betrachtungsebene and Brennweite bestimmt werden. Das objektorientierte Graphiksystem besteht aus einer Bibliothek *grid* with den notwendigen elementaren Operationen, and einer darauf aufbauende Bibliothek *lattice*, die die aus der Basis-Graphik bekannten Darstellungen neu implementiert and durch weitere ergänzt.

Lattice-objects werden with *print()* ausgegeben.

xpl:ch04-wireframe

**Example 4.2:**

```
library(lattice)
## volcano ## 87 x 61 matrix
print(wireframe(volcano, shade = TRUE,
                aspect = c(61/87, 0.4),
                light.source = c(10,0,10)))
```



Basis-Graphik and Lattice-Graphik are getrennte Graphik-Systeme. Leider benutzen sie auch vergleichbare Functions unterschiedliche Bezeichnungen, und vergleichbare Displays haben unterschiedliche Darstellungen. Eine kleine Übersetzungshilfe ist in Tabelle 4.4 angegeben. Einige Hilfsfunktionen, um beide Graphik-Systeme in Kombination zu nutzen, werden durch die Bibliothek `gridBase` bereitgestellt. Eine ausführliche Einführung in beide Graphik-System ist [T3].

for Visualisierungen im weiten Spektrum von wissenschaftlichen Visualisierungen bis hin zu aufwendigen Spielen wird verbreitet OpenGL benutzt. Dessen Functions stehen auch in R durch die Bibliothek `rgl` zur Verfügung. Es gibt jedoch einen deutlichen Unterschied zwischen den üblichen Anforderungen an Graphik, und den speziellen Erfordernissen statistischer Graphik. Wenn es um die Darstellungen von Functions geht, ist statistische Graphik noch vergleichbar mit den Anforderungen der üblichen Analysis. Der kleine Unterschied ist, dass Functions in der Statistik häufig stückweise konstant or nur stückweise stetig are, während z.B. in der Analysis stetige or sogar in differenzierbare Functions die Regel are. Bei der Darstellung von Daten ändert sich die Situation drastisch. Statistische Daten are üblicherweise diskret. Glattheitseigenschaften, die die Visualisierung analytischer Daten einfacher machen, fehlen bei statistischen Daten. Deshalb are spezielle angepasste Visualisierungen nötig.

**To Do:**  
discuss  
alpha  
channel

Basis-Graphik		Lattice
<code>barplot()</code>	bar chart	<code>barchart()</code>
<code>boxplot()</code>	box and whisker plot	<code>bwplot()</code>
	3 dimensional scatter plot	<code>cloud()</code>
<code>contour</code>	contour plot	<code>contourplot()</code>
<code>coplot</code>	conditional scatter plots	<code>contourplot()</code>
<code>curve(density())</code>	density estimator	<code>densityplot()</code>
<code>dotchartt()</code>	dot plot	<code>dotplot()</code>
<code>hist()</code>	dot plot	<code>histogram()</code>
<code>image()</code>	colour map plots	<code>splom()</code>
	parallel coordinate plots	<code>parallel()</code>
<code>pairs()</code>	scatter plot matrices	<code>wireframe()</code>
<code>persp()</code>	three dimensional surface	<code>wireframe()</code>
<code>plot()</code>	scatter plot	<code>xyplot()</code>
<code>qqnorm()</code>	theoretical $Q - Q$ -plot	<code>qqmath()</code>
<code>qqplot()</code>	empirical $Q - Q$ -plot	<code>qq()</code>
<code>stripchart()</code>	one dimensional scatterplot	<code>stripplot()</code>

Table 4.4 *Basis-Graphik and Lattice*

ta:ch04lattice

## 4.2 Dimensions

Wenn we von einer Dimension zu höheren Dimensionen gehen, gibt es sowohl for die theoretische Untersuchung als auch for die grafische Darstellung neue Herausforderungen. The linearen Modelle können wieder als leitendes or warnendes Beispiel dienen.

The Herausforderungen können von ernsthaften Problemen stammen. So können Verteilungen auf höherdimensionalen Räumen selbst unter Regularitätsvoraussetzungen unüberschaubar komplex sein. The Klassifikations- and Identifikationsprobleme for Functions and Räume aus Analysis and Geometrie geben einen Vorgeschnack davon, was bei der Untersuchung von probabiliyverteilungen zu bewältigen is.

Daneben gibt es hausgemachte Probleme: Eigentore, die durch selbstgetroffene Wahlen erst erzeugt werden.

Ein Beispiel for hausgemachte Probleme kann an linearen Modellen illustriert werden. The Interpretation des Gauß-Markoff-Schätzers als lineare Projektion zeigt, dass nur scheinbar Koeffizienten for einzelne Regressoren geschätzt werden. Eigentlich wird ein vector im von den Regressoren aufgespannten Raum geschätzt; die Zuordnung zu den einzelnen Regressoren is dann nur lineare Algebra. Diese hängt nicht von dem Einfluss des einzelnen Regressors ab, sondern von der gemeinsamen Geometrie der Regressoren. Nur wenn die Regressoren eine Orthogonalbasis bilden, gibt es eine direkte Interpretation der Koeffizienten. Wird im linearen

Modell die Liste der Regressoren z.B. dupliziert, so ändert sich der Raum nicht. The Rechnungen in Koordinaten werden etwas komplizierter, weil die Regressoren nun auf keinen Fall eine Basis bilden, aber von a abstrakten Standpunkt bleibt die Situation unverändert. Gibt es aber kein echtes Duplikat, sondern geringfügige Abweichungen (durch minimale "Fehler", Rundungen, Transformationen), so ändert sich die Situation drastisch. for den Gauß-Markoff-Schätzer is nur der von den Regressoren aufgespannte Raum relevant, and selbst durch minimale Änderungen im Duplikat kann sich dessen Dimension verdoppeln. Dies is ein Beispiel for ein hausgemachtes Problem.

Dies and andere examples are ein Grund, die Beziehungen zwischen den Variablen genauer zu untersuchen. Bei der Regression etwa betrifft dies nicht nur die Beziehung zwischen Respons and Regressor, sondern, wie durch das letzte Beispiel illustriert, auch die Beziehungen zwischen den Regressoren.

Um die Verbindung zu den Regressionsproblemen zu halten and auf die Erfahrungen in diesem Bereich zurückzugreifen, betten we formal die Regressionsprobleme in einen allgemeineren Rahmen ein. Bei der Regression hatten we eine herausgehobene Variable, die Respons  $Y$ , deren distribution in Abhängigkeit von den valuesn der übrigen Variablen, der Regressoren  $X$ , modelliert werden sollte. We fassen jetzt Respons and Regressor zu a Datenvektor  $Z = (Y; X)$  zusammen and werden auch die gemeinsame distribution von Z diskutieren. We finden das Regressionsproblem in diesem allgemeineren Rahmen wieder: beim Regressionsproblem suchten we nach a Schätzer for die Mittelwertsfunktion  $m$  im Modell

$$Y = m(X) + \varepsilon.$$

Im allgemeineren Rahmen berücksichtigen we eine gemeinsame distribution von  $X$  and  $Y$ . Das Regressionsmodell wird damit zum Modell

$$Y = E(Y|X) + \varepsilon$$

and we haben zunächst die Identifizierung  $m(X) = E(Y|X)$ .

Wenn we tatsächlich am ursprünglichen Regressionsmodell interessiert are, müssen we weitere Arbeit leisten. Eine Schätzung des bedingten Erwartungswerts  $E(Y|X)$  is nicht dasselbe wie die Schätzung einer Regressionsfunktion  $m(X)$ . Bei dem Regressionsproblem haben we keine Annahmen über die distribution von  $X$  gemacht. Um von  $E(Y|X)$  (or a Schätzer dafür) auf  $m(X)$  zurück zu schließen, müssen we überprüfen, dass die Schätzung von Verteilungsannahmen über  $X$  unabhängig is. for unsere augenblicklichen Zwecke is diese Unterscheidung aber nicht relevant. We können uns eine Ignoranz auf Zeit erlauben.

Der allgemeine Rahmen in diesem Kapitel is also:

we untersuchen Daten  $(Z_i)_{i=1,\dots,n}$ , wobei die einzelnen observations values in  $\mathbb{R}^q$  annehmen.

Haben we im wesentlichen lineare Strukturen, so können we oft auch höher-dimensionale Strukturen with Methoden analysieren, die for eindimensionale Modelle entwickelt are. We müssen die Methoden evtl. modifizieren or iteriert anwenden. Sie helfen uns jedoch, die wesentlichen Merkmale zu erkennen. Sie versagen jedoch, wenn sich höhere Dimensionalität with Nichtlinearität verbindet. Dann are speziellere Methoden gefragt.

### 4.3 Selections

Ursprünglich bedeutet eine Selektion eine Auswahl von observations. for die grafische Darstellung wird die Selektion with einer Ausprägung von Attributen (z.B. Farbe, Plot-Zeichen, Dicke)

`linking|textbf` assoziiert. Alle Variablenvalues, die zu observations in der Selektion gehören, werden with die-  
`brushing|textbf` sen Attributen in dieser Ausprägung dargestellt. Dies ermöglicht es, die Verbindung (“**linking**”) der Selektion in verschiedenen Plots zu verfolgen. So können Selektionen helfen, Strukturen in verbundenen Plots, zu erkennen.

In der praktischen Datenanalyse werden die Selektionen variiert (“**brushing**”), um observations zu zusammengehörigen Beobachtungsgruppen zusammen zu fassen. Dies ist ein wichtiges Werkzeug der interaktiven Datenanalyse. Selektionen werden, statistisch gesprochen, zur Modellwahl benutzt. Ihnen entspricht das Konzept der lokalen Modelle: anstelle ein for die Daten globales, möglicherweise sehr komplexes Modell zu benutzen, werden for jede Selektion möglicherweise einfachere Modelle bestimmt, die jeweils nur for die observations aus dieser Selektion gelten.

Das Linking wird leider von R nicht direkt unterstützt. We müssen also jeweils selbst sicher stellen, dass Selektionen with den entsprechenden Attributen dargestellt werden. Auch die Repräsentation von Selektionen is in R nicht einheitlich. Bei Funktionsaufrufen können diese durch `selection`-parameter realisiert sein, or durch `group`-Variable, or als Bedingung in a Formelausdruck. Deshalb müssen we uns in jedem Fall with ad-hoc-Lösungen begnügen.

R bleibt weitgehend auf statische Selektionen beschränkt, so dass Brushing nur rudimentär möglich is.

Selektionen werden im Zusammenhang bei den nachfolgenden examplesn illustriert.

#### 4.4 Projections

ch04projektionen

Als erstes Beispiel betrachten we einen Datensatz aus einer Arbeit ([20]), in der unterschiedliche Diabetes-Arten untersucht worden. Der Datensatz is zum Beispiel in `library(locfit)` verfügbar. The Variablen umfassen Laborvalues zum Glukose-Stoffwechsel and are in Tabelle 4.5 erklärt.

```
library(locfit)
data(chemdiab)
```

Eine erste Übersicht erhalten we with

```
summary(chemdiab)
```

Output			
rw	fpg	ga	ina
Min. : 0.7100	Min. : 70.0	Min. : 269.0	Min. : 10.0
1st Qu.: 0.8800	1st Qu.: 90.0	1st Qu.: 352.0	1st Qu.: 118.0
Median : 0.9800	Median : 97.0	Median : 413.0	Median : 156.0
Mean : 0.9773	Mean : 122.0	Mean : 543.6	Mean : 186.1
3rd Qu.: 1.0800	3rd Qu.: 112.0	3rd Qu.: 558.0	3rd Qu.: 221.0
Max. : 1.2000	Max. : 353.0	Max. : 1568.0	Max. : 748.0
sspg	cc		
Min. : 29.0	Chemical_Diabetic:36		
1st Qu.: 100.0	Normal : 76		
Median : 159.0	Overt_Diabetic : 33		

name	Variable	Einheit, Bem.	Marginalverteilung   textbf Scatterplot- Matrix   textbf pairs@pairs   textit pairs@pairs   textbf Topic hplot! pairs@pairs
rw	relatives Gewicht		
fpg	Plasma-Glukose (nach Fasten)	[mg/100 ml]	
ga	Glukosespiegel integriert über 3 Stunden Toleranztest	[mg/100 ml × h]	
ina	Insulinspiegel integriert über 3 Stunden Toleranztest	[μU/100 ml × h]	
sspg	Plasma-Glukose (steady state)	[mg/100 ml]	
cc	Klassifikation	chemische, normale, offene Diabetes	

Table 4.5 *Diabetes-Datensatz: Variable***diabvars**

Mean : 184.2  
 3rd Qu.: 257.0  
 Max. : 480.0

Wie in der Originalarbeit lassen wir das relative Gewicht außer Betracht. The chemische Klassifikation *cc* is aus den Stoffwechseldaten abgeleitet. Sie beinhaltet also keine eigene Information. Zur Orientierung benutzen wir sie dennoch als Markierung, d.h. wir benutzen die Selektion *cc* = *Chemical\_Diabetic*, *Normal*, *Overt\_Diabetic*. Der Kern des Datensatzes is vierdimensional with den Variablen *fpg*, *ga*, *ina*, *sspg*.

#### 4.4.1 Marginal Distributions and Scatterplot Matrices

We können versuchen, die mehrdimensionale distribution zu untersuchen, indem wir uns die zweidimensionalen **Marginalverteilungen** (Randverteilungen) for alle Variablenpaare ansehen. The grafische Darstellung dazu heißt **Scatterplot-Matrix**, in R als function *pairs()* implementiert.

**help(pairs)**

**pairs** *Scatterplot Matrices*

#### Description

A matrix of scatterplots is produced.

```
data.matrix@data.matrix|textit
Usage
pairs(x, ...)

## S3 method for class 'formula':
pairs(formula, data = NULL, ..., subset,
      na.action = stats::na.pass)

## Default S3 method:
pairs(x, labels, panel = points, ...,
      lower.panel = panel, upper.panel = panel,
      diag.panel = NULL, text.panel = textPanel,
      label.pos = 0.5 + has.diag/3,
      cex.labels = NULL, font.labels = 1,
      row1attop = TRUE, gap = 1)
```

### Arguments

<b>x</b>	the coordinates of points given as numeric columns of a matrix or data frame. Logical and factor columns are converted to numeric in the same way that <code>data.matrix</code> does.
<b>formula</b>	a formula, such as <code>~ x + y + z</code> . Each term will give a separate variable in the pairs plot, so terms should be numeric vectors. (A response will be interpreted as another variable, but not treated specially, so it is confusing to use one.)
<b>data</b>	a <code>data.frame</code> (or list) from which the variables in <b>formula</b> should be taken.
<b>subset</b>	an optional vector specifying a subset of observations to be used for plotting.
<b>na.action</b>	a function which indicates what should happen when the data contain NAs. The default is to pass missing values on to the panel functions, but <code>na.action = na.omit</code> will cause cases with missing values in any of the variables to be omitted entirely.
<b>labels</b>	the names of the variables.
<b>panel</b>	<code>function(x,y,...)</code> which is used to plot the contents of each panel of the display.
<b>...</b>	arguments to be passed to or from methods. Also, graphical parameters can be given as can arguments to <code>plot</code> such as <code>main</code> . <code>par("oma")</code> will be set appropriately unless specified.
<b>lower.panel, upper.panel</b>	separate panel functions to be used below and above the diagonal respectively.
<b>diag.panel</b>	optional <code>function(x, ...)</code> to be applied on the diagonals.
<b>text.panel</b>	optional <code>function(x, y, labels, cex, font, ...)</code> to be applied on the diagonals.
<b>label.pos</b>	y position of labels in the text panel.
<b>cex.labels, font.labels</b>	graphics parameters for the text panel.

<code>row1attop</code>	logical. Should the layout be matrix-like with row 1 at the top, or graph-like with row 1 at the bottom?
<code>gap</code>	Distance between subplots, in margin lines.

### Details

The  $ij$ th scatterplot contains `x[, i]` plotted against `x[, j]`. The “scatterplot” can be customised by setting panel functions to appear as something completely different. The off-diagonal panel functions are passed the appropriate columns of `x` as `x` and `y`: the diagonal panel function (if any) is passed a single column, and the `text.panel` function is passed a single (`x`, `y`) location and the column name.

The graphical parameters `pch` and `col` can be used to specify a vector of plotting symbols and colors to be used in the plots.

The graphical parameter `oma` will be set by `pairs.default` unless supplied as an argument. A panel function should not attempt to start a new plot, but just plot within a given coordinate system: thus `plot` and `boxplot` are not panel functions.

By default, missing values are passed to the panel functions and will often be ignored within a panel. However, for the formula method and `na.action = na.omit`, all cases which contain a missing values for any of the variables are omitted completely (including when the scales are selected). (The latter was the default behaviour prior to R 2.0.0.)

### Author(s)

Enhancements for R 1.0.0 contributed by Dr. Jens Oehlschlaegel-Akiyoshi and R-core members.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### Examples

```

pairs(iris[1:4], main = "Anderson's Iris Data -- 3 species",
      pch = 21, bg = c("red", "green3", "blue") [unclass(iris$Species)])
```

```

## formula method
pairs(~ Fertility + Education + Catholic, data = swiss,
      subset = Education < 20, main = "Swiss data, Education < 20")
```

```

pairs(USJudgeRatings)
```

```

## put histograms on the diagonal
panel.hist <- function(x, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)

```

4-10

DIMENSIONS 1, 2, 3, ...,  $\infty$

```
breaks <- h$breaks; nB <- length(breaks)
y <- h$counts; y <- y/max(y)
rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)
}
pairs(USJudgeRatings[1:5], panel=panel.smooth,
      cex = 1.5, pch = 24, bg="light blue",
      diag.panel=panel.hist, cex.labels = 2, font.labels=2)

## put (absolute) correlations on the upper panels,
## with size proportional to the correlations.
panel.cor <- function(x, y, digits=2, prefix="", cex.cor)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits=digits)[1]
  txt <- paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex * r)
}
pairs(USJudgeRatings, lower.panel=panel.smooth, upper.panel=panel.cor)
```

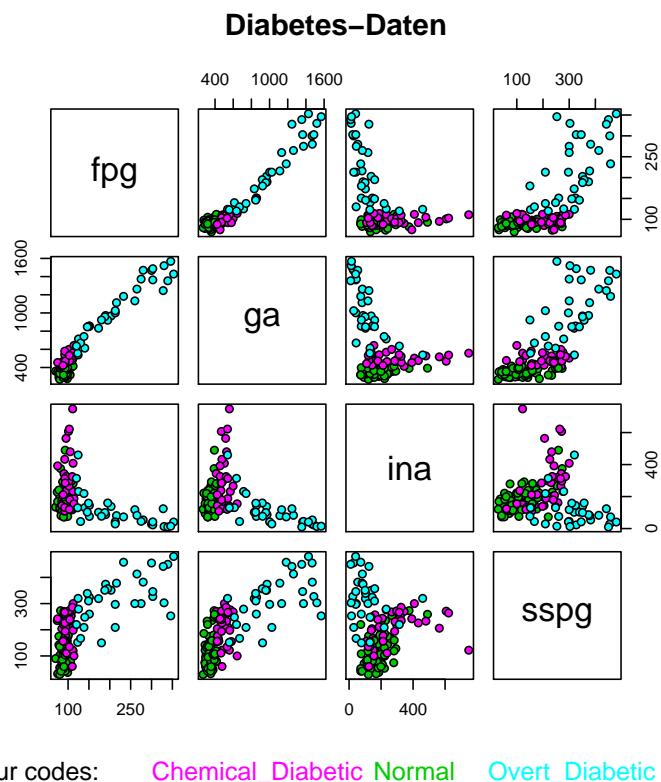
We benutzen die chemischen Diabetes-Klassen `cc` als Selektionen. Jeder dieser Selektionen wird ein Farbwert zugeordnet; dies ist das verbindende Attribut, das ermöglicht, Verbindungen zwischen den Plots zu verfolgen. Um diese Verbindung zu dokumentieren, müssen wir in die Grafiksteuerung eingreifen und die Plots modifizieren. Wir erzeugen mit dem Parameter `oma` einen äusseren Rand, in dem wir eine Legende platzieren.

expl:ch04-diabetes

*Example 4.3:*

```
pairs(~fpg + ga + ina + sspg, data = chemdiab, pch = 21,
      main = "Diabetes-Daten",
      bg = c("magenta", "green3", "cyan")[unclass(chemdiab$cc)],
      oma = c(8, 8, 8, 8))
mtext(c("Colour codes:", levels(chemdiab$cc)),
      col = c("black", "magenta", "green3", "cyan"),
      at = c(0.1, 0.4, 0.6, 0.8), side = 1, line = 2)
```

pairs@pairs|textit



The function `pairs()` kontrolliert nur das “Layout” der Matrix, die Auswahl und Anordnung der Projektionen. The Darstellung in den Plot-Feldern kann durch den Aufruf gesteuert werden. The Default-Belegungen führen dazu, dass in der Diagonale die names der Variablen und außerhalb der Diagonalen die paarweisen Scatterplots gezeigt werden.

a:ch04:4

**ToDo:**  
 supply  
 better  
 support  
 for legend,  
 colour key

<pre>print@print te cloud@cloud te</pre>	<b>Aufgabe 4.1</b> <p>Generieren Sie eine Scatterplot-Matrix für den Diabetes-Datensatz, die in der Diagonale die Histogramme der jeweiligen Variablen zeigt.</p> <p><i>Hinweis:</i> Siehe <code>help(pairs)</code>.</p>
--	--

Bestimmte Aspekte der distribution können aus den Randverteilungen einfach abgelesen werden. Andere geometrische Strukturen are aus Randverteilungen gar nicht or nur schwer zu rekonstruieren.

Zwischen dem Glukose-Spiegel bei Fasten `fpg` and dem integrierten Glukose- Spiegel bei Belastung `ga` besteht z.B. ein deutlicher linearer Zusammenhang. Dieser is in den zweidimensionalen Marginalverteilungen erkennbar and kann with den Methoden for lineare Modelle untersucht werden.

Diese deutliche Beziehung pflanzt sich auf die Beziehungen zu den anderen Variablen `ina`, `sspg` fort. In der Originalarbeit wird deshalb `fpg` nicht weiter berücksichtigt. Zu untersuchen are noch die Variablen `ga`, `ina`, `sspg`. The dreidimensionale Struktur dieses Teils des Datensatzes is aus den Marginalverteilungen nicht einfach abzulesen.

#### 4.4.2 Projection Pursuit

Geometrische Beziehungen or stochastische Abhängigkeiten, die nicht parallel zu den Koordinaten-Achsen ausgerichtet are, werden durch die Randverteilungen nicht ausgedrückt. We können die Idee verallgemeinern and anstelle von zweidimensionale Marginalverteilungen beliebige Projektionen benützen. Dazu greifen we auf `library(lattice)` zu. Darin is ein an einer Kamera orientiertes Grafik-Modell implementiert.

The `grid`-Grafik liefert with dem Paket `lattice` eine weitgehende Unterstützung for multivariate Darstellungen. `grid` is dabei die Basis. Das ursprüngliche Grafiksystem von R implementiert ein Modell, dass an der Vorstellung von Stift and Papier orientiert is. Ein Grafik-Port (Papier) wird eröffnet and darauf werden Linien, Punkte/Symbole gezeichnet. `grid` is ein zweites Grafiksystem, das an a Kamera/Objekt-Modell orientiert is. Grafische objects in unterschiedlicher Lage and Richtung werden in a visuellen Raum abgebildet. Auf der `grid` baut `lattice` auf. In <<http://cm.bell-labs.com/cm/ms/departments/sia/project/trellis/>> are die Grundideen zur Visualisierung multidimensionaler Daten dokumentiert, die in `lattice` implementiert are.

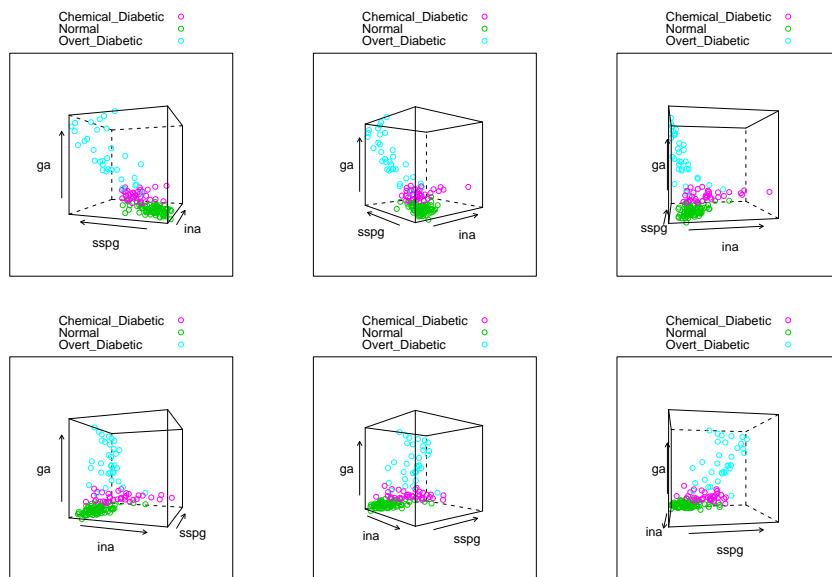
The erzeugte Grafik wird with `print()` ausgegeben. Mit dem parameter `split` können we den Ausgabebereich aufteilen. Leider is das Linking hier gebrochen: `cloud()` kann zwar eine Legende erzeugen, diese Zeigt jedoch die Farbskala bei Beginn der Grafik, nicht die bei der Ausgabe benutzte. We müssen deshalb wieder ins System [Input](#) greifen and diesmal die Farbtabellen ändern.

```
library("lattice")
diabcloud <- function(y, where, more = TRUE, ...) {
  print(cloud(ga ~ ina + sspg, data = chemdiab, groups = cc,
              screen = list(x = -90, y = y), distance = .4, zoom = .6,
              auto.key = TRUE, ...),
        split = c(where, 3, 2), more = more)
}
supsym <- trellis.par.get("superpose.symbol")
```

```

supsymold <- supersym
supsym$col = c("magenta", "green3", "cyan")
trellis.par.set("superpose.symbol" = supersym)
diabcloud(y = 70, where = c(1, 1))
diabcloud(y = 40, where = c(2, 1))
diabcloud(y = 10, where = c(3, 1))
diabcloud(y = -20, where = c(1, 2))
diabcloud(y = -50, where = c(2, 2))
diabcloud(y = -80, where = c(3, 2), more = FALSE)
trellis.par.set("superpose.symbol" = supersymold)
rm(diabcloud, supersymold, supersym)

```



a:ch04:1

Aufgabe 4.2	
	<p>Modifizieren Sie dieses Beispiel so, dass Sie einen Eindruck der dreidimensionalen Struktur bekommen.</p> <p>Wie unterscheidet sich offene Diabetes von chemischer Diabetes?</p> <p>Wie verhält sich die Normal-Gruppe zu den beiden Diabetes-Gruppen?</p>

Auch with Serien von Projektionen is es oft nicht einfach, eine dreidimensionale Struktur zu identifizieren. Mit animierten Folgen kann dies einfacher sein. Unterstützung dazu findet sich in `library(rggobi)`, die allerdings ggobi, zu finden in <http://www.ggobi.org/>, als zusätzliche Software voraussetzt.

Was hier ad-hoc gemacht wird, kann auch systematisch durchgeführt werden and for beliebige Dimensionen verallgemeinert werden: man sucht for einen Datensatz im  $\mathbb{R}^q$  nach “interessanten”

**projection pursuit** Projektionen. Dazu definiert man einen Index, der messen soll, wie interessant eine Projektion ist, and lässt dann eine Suche laufen, die diesen Index maximiert. The auf dieser Idee basierende Familie statistischer Verfahren findet man unter dem Stichwort **projection pursuit**. Das System **ggobi** beinhaltet Implementierungen von projection pursuit for eine Reihe von Indizes, die über die Functions in `library(rggobi)` von R aus angesprochen werden können.

#### 4.4.3 Projections for Dimensions 1, 2, 3, ... 7

Projektionsmethoden versuchen, in a höherdimensionalen Datensatz Strukturen niedrigerer Dimension zu identifizieren. The identifizierbare Dimension is dabei beschränkt. Projizieren we objects with einer Dimension, die größer is als das Projektionsziel, so überdeckt die typische Projektion alles, gibt also keine Information mehr.

Wieviele Dimension können we erfassen? The grafische Darstellung in der Ebene gibt zunächst eine niedrig angesetzte Grenze von zwei Dimensionen, d.h. zweidimensionale Strukturen können we direkt with cartesischen Koordinaten in der *xy*-Ebene darstellen. The Wahrnehmung kann dreidimensionale Strukturen anhand von Hinweisen auf die Raumtiefe (etwa durch Schatten) or aus Folgen von 2d-Bildern rekonstruieren. Mit Animationen erhalten we einen Eindruck von veränderlichen 3d-Folgen and are damit bei vier Dimensionen.

Mit zusätzlichen Informationskanälen wie z.B. with Farbcodierungen können we dies leicht erhöhen, bleiben aber effektiv bei vier bis sieben Dimensionen for ein Display.

The Kombination mehrerer Displays hilft kaum über diese Grenze hinaus. Stellen we mehrere Displays z.B. in einer Scatterplot-Matrix dar, so verlieren we die Fähigkeit, for die einzelnen Szenen durch die Wahrnehmung komplexere Strukturen zu generieren. Anstelle dessen müssen we aktiv durch Vergleichen die komplexeren Strukturen aus den zweidimensionalen Displays erarbeiten. The Fähigkeit, simultane Vergleiche durchzuführen, is dabei beschränkt. Ebenso die Anzahl von Displays, die simultan auf a Seiten-Medium wie Bildschirm or Papier dargestellt werden kann.

#### ToDo:

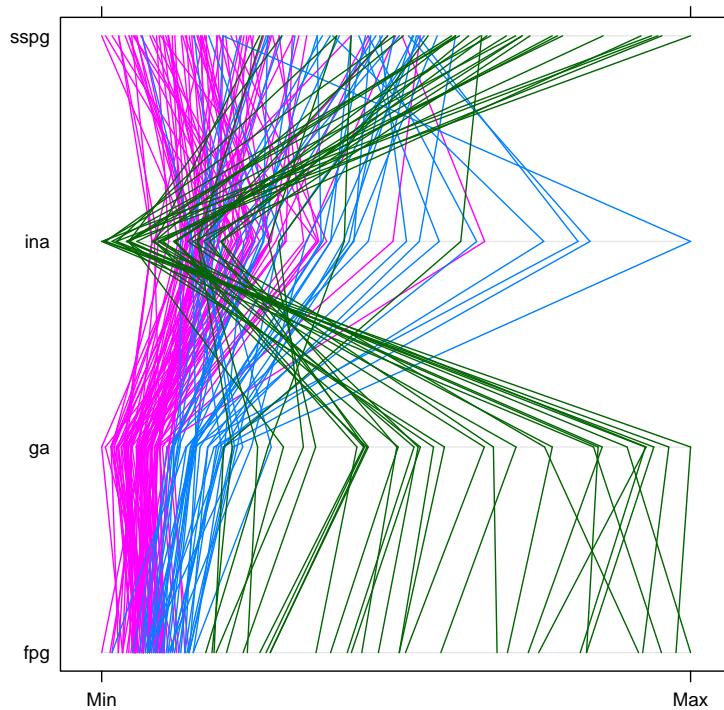
Does memory help to increase the bound?

#### 4.4.4 Parallel Coordinates

The grafische Darstellung (in kartesischen Koordinaten) are zunächst auf ein- and zweidimensionale Projektionen beschränkt. Aber selbst bei Darstellungen in der Ebene is die Beschränkung auf zwei Dimensionen nicht vorgegeben, sondern is eine sequence unserer Wahl der Darstellung in kartesischen Koordinaten. Plot-Matrizen durchbrechen diese Dimensionsschranke durch Kombination von kartesischen Koordinatensystemen.

Parallel-Koordinaten orientieren die Achsen for die Variablen parallel zueinander. for Häufigkeiten bei kategorialen Variablen is dies eine übliche Darstellung: (evtl. überlagerte) Balkendiagramme benutzen Parallel-Koordinaten. The Prozedur `parallel()` in `library(lattice)` unterstützt Parallel-Koordinaten auch for quantitative Variable. The zu a Fall gehörenden Markierungen auf diesen Achsen werden durch `ins187$pcord` verbunden. Diese Form der Parallel-Koordinaten stammt von A. Inselberg [9].

```
library("lattice")
print(parallel(chemdiab[2:5], groups = chemdiab$cc))
```



The Information is dieselbe wie in den vorausgehenden Grafiken. Durch die veränderte Darstellung werden die Zusammenhänge auf neue Weise zugänglich.

a:ch04:2

**To Do:**  
add  
legend,  
control  
colour

Aufgabe 4.3	
	Notieren Sie für den <code>chemdiab</code> -Datensatz (schriftlich!) die Beziehungen zwischen den Variablen, die Sie im Parallel-Koordinatenplot erkennen können.
	Anstelle von <code>chemdiab[2:5]</code> können Sie die Variablen auch explizit als <code>chemdiab[c(2, 3, 4, 5)]</code> angeben. Durch dieser Form erhalten Sie Kontrolle über die Reihenfolge der Variablen. Vergleichen Sie zwei unterschiedliche Anordnungen der Variablen und notieren Sie (schriftlich!) ihre observations. Welche Variablen-Anordnung gibt die einfachere Darstellung? Welche Beziehungen zwischen den Variablen sind in beiden ablesbar? Welche nur in einer der Anordnungen?

**bedingt | textbf 4.5 Sections, Conditional Distributions and Coplots**

Coplot | textbf 4.5  
 coplot@coplot | textbf  
 co.intervals@co.intervals  
 (coplot) Schnitte are, abstrakt gesehen, bedingte distributions des Typs  $P(\cdot \mid X = x)$ . Sie are nur dort zuverlässig, wo der Schnitt eine Bedingung definiert, die ein positives Maß hat. Um die hplot!coplot Reduktion auf bedingte distributions auch auf Daten anwenden zu können, dicken Topic we die Schnitte auf. Anstelle bedingter distributions des Typs  $P(\cdot \mid X = x)$  zu untersuchen, Topic **aplot!coplot** testen we  $P(\cdot \mid \|X - x\| < \varepsilon)$ , wobei  $\varepsilon$  auch with  $x$  variieren kann. In grafischen Darstellungen von Daten verlangt dies eine Serie von Plots, die jeweils nur den durch die Bedingung eingeschränkten Teildatensatz zeigen.

Statistisch führen Projektionen zu Marginalverteilungen and Schnitte zu bedingten distributions. Schnitte and Projektionen are in gewissem Sinne komplementär: Projektionen zeigen Strukturmerkmale niedriger Dimension. Schnitte are geeignet, Strukturmerkmale niedriger Co-Dimension zu entdecken. Beide können zur Datenanalyse kombiniert werden. Das Wechselspiel von Projektionen and Schnitten is in [5] untersucht.

Wie die Dimensionsgrenzen bei der Projektion gibt es Grenzen for die Co-Dimension bei den Schnitten. We können nur objects kleiner Co-Dimension erfassen. Ist die Co- Dimension zu groß, so is ein typischer Schnitt leer, gibt also keine Information.

Als erstes Hilfsmittel stellt R die Möglichkeit bereit, zwei Variablen **bedingt** auf eine or mehrere weitere Variable zu analysieren. Als grafische Darstellung dient dazu der **Coplot**. Er is eine Variante der Plot-Matrix and zeigt in jedem Feld den Scatterplot zweier Variabler, gegeben die Bedingung.

Der Coplot kann nun auf bestimmte Muster untersucht werden. Sind die dargestellten Variablen stochastisch unabhängig von den bedingenden Variablen, so zeigen alle Plot-Elemente dieselbe Gestalt. Dargestellte Variable and bedingende Variable können dann entkoppelt werden.

Stimmt die Gestalt überein, aber Ort and Größe variieren, so weist dies auf eine (nicht notwendig lineare) Shift/Skalenbeziehung hin. Additive Modelle or Varianten davon können benutzt werden, um die Beziehung zwischen dargestellten Variablen and bedingenden Variablen zu modellieren.

Verändert sich bei Variation der Bedingung die Gestalt, so liegt eine wesentliche Abhängigkeitsstruktur or Interaktion vor, die genauerer Modellierung bedarf.

**help(coplot)**

coplot

*Conditioning Plots*

*Description*

This function produces two variants of the **conditioning plots** discussed in the reference below.

*Usage*

```
coplot(formula, data, given.values, panel = points, rows, columns,
       show.given = TRUE, col = par("fg"), pch = par("pch"),
       bar.bg = c(num = gray(0.8), fac = gray(0.95)),
       xlab = c(x.name, paste("Given :", a.name)),
       ylab = c(y.name, paste("Given :", b.name)),
       subscripts = FALSE,
       axlabels = function(f) abbreviate(levels(f)),
       number = 6, overlap = 0.5, xlim, ylim, ...)
co.intervals(x, number = 6, overlap = 0.5)
```

*Arguments*

<b>formula</b>	a formula describing the form of conditioning plot. A formula of the form $y \sim x   a$ indicates that plots of $y$ versus $x$ should be produced conditional on the variable $a$ . A formula of the form $y \sim x   a * b$ indicates that plots of $y$ versus $x$ should be produced conditional on the two variables $a$ and $b$ .
<b>data</b>	All three or four variables may be either numeric or factors. When $x$ or $y$ are factors, the result is almost as if <code>as.numeric()</code> was applied, whereas for factor $a$ or $b$ , the conditioning (and its graphics if <code>show.given</code> is true) are adapted.
<b>given.values</b>	a data frame containing values for any variables in the formula. By default the environment where <code>coplot</code> was called from is used.
<b>panel</b>	a value or list of two values which determine how the conditioning on $a$ and $b$ is to take place.
<b>rows</b>	When there is no $b$ (i.e., conditioning only on $a$ ), usually this is a matrix with two columns each row of which gives an interval, to be conditioned on, but it can also be a single vector of numbers or a set of factor levels (if the variable being conditioned on is a factor). In this case (no $b$ ), the result of <code>co.intervals</code> can be used directly as <code>given.values</code> argument.
<b>columns</b>	a function( $x, y, col, pch, \dots$ ) which gives the action to be carried out in each panel of the display. The default is <code>points</code> .
<b>show.given</b>	the panels of the plot are laid out in a <code>rows</code> by <code>columns</code> array. <code>rows</code> gives the number of rows in the array.
<b>col</b>	the number of columns in the panel layout array.
<b>pch</b>	logical (possibly of length 2 for 2 conditioning variables): should conditioning plots be shown for the corresponding conditioning variables (default <code>TRUE</code> )
<b>bar.bg</b>	a vector of colors to be used to plot the points. If too short, the values are recycled.
	a vector of plotting symbols or characters. If too short, the values are recycled.
	a named vector with components "num" and "fac" giving the background colors for the (shingle) bars, for <b>numeric</b> and <b>factor</b> conditioning variables respectively.

<code>factor@factor textit</code>	<code>xlab</code>	character; labels to use for the x axis and the first conditioning variable. If only one label is given, it is used for the x axis and the default label is used for the conditioning variable.
<code>par@par textit</code>	<code>ylab</code>	character; labels to use for the y axis and any second conditioning variable.
<code>mtext@mtext textit</code>	<code>smooth@smooth textit</code>	logical: if true the panel function is given an additional (third) argument subscripts giving the subscripts of the data passed to that panel.
<code>title@title textit</code>	<code>panel.smooth@panel smooth textit</code>	function for creating axis (tick) labels when x or y are factors.
<code>matrix@matrix textit</code>	<code>points@points textit</code>	integer; the number of conditioning intervals, for a and b, possibly of length 2. It is only used if the corresponding conditioning variable is not a factor.
<code>range@range textit</code>		numeric < 1; the fraction of overlap of the conditioning variables, possibly of length 2 for x and y direction. When overlap < 0, there will be gaps between the data slices.
<code>pairs@pairs textit</code>		the range for the x axis.
<code>panel.smooth@panel smooth textit</code>		the range for the y axis.
<code>points@points textit</code>		additional arguments to the panel function.
		<code>x</code> a numeric vector.

### Details

In the case of a single conditioning variable `a`, when both `rows` and `columns` are unspecified, a “close to square” layout is chosen with `columns >= rows`.

In the case of multiple `rows`, the *order* of the panel plots is from the bottom and from the left (corresponding to increasing `a`, typically).

A panel function should not attempt to start a new plot, but just plot within a given coordinate system: thus `plot` and `boxplot` are not panel functions.

As from R 2.0.0 the rendering of arguments `xlab` and `ylab` is not controlled by `par` arguments `cex.lab` and `font.lab` even though they are plotted by `mtext` rather than `title`.

### Value

`co.intervals(., number, .)` returns a (`number`  $\times$  2) `matrix`, say `ci`, where `ci[k,]` is the range of x values for the k-th interval.

### References

- Chambers, J. M. (1992) *Data for models*. Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.  
 Cleveland, W. S. (1993) *Visualizing Data*. New Jersey: Summit Press.

### See Also

`pairs`, `panel.smooth`, `points`.

*Examples*

```

## Tonga Trench Earthquakes
coplot(lat ~ long | depth, data = quakes)
given.depth <- co.intervals(quakes$depth, number=4, overlap=.1)
coplot(lat ~ long | depth, data = quakes, given.v=given.depth, rows=1)

## Conditioning on 2 variables:
ll.dm <- lat ~ long | depth * mag
coplot(ll.dm, data = quakes)
coplot(ll.dm, data = quakes, number=c(4,7), show.given=c(TRUE,FALSE))
coplot(ll.dm, data = quakes, number=c(3,7),
       overlap=c(-.5,.1)) # negative overlap DROPS values

## given two factors
Index <- seq(length=nrow(warpbreaks)) # to get nicer default labels
coplot(breaks ~ Index | wool * tension, data = warpbreaks, show.given = 0:1)
coplot(breaks ~ Index | wool * tension, data = warpbreaks,
       col = "red", bg = "pink", pch = 21, bar.bg = c(fac = "light blue"))

## Example with empty panels:
with(data.frame(state.x77), {
  coplot(Life.Exp ~ Income | Illiteracy * state.region, number = 3,
         panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...))
  ## y ~ factor -- not really sensical, but 'show off':
  coplot(Life.Exp ~ state.region | Income * state.division,
         panel = panel.smooth)
})

```

---

We illustrieren die Coplots with dem “Quakes”-Datensatz. Dieser Datensatz gibt die geografische Länge und Breite einer Reihe von Erdbeben in der Nähe der Fiji-Inseln, zusammen with der Tiefe des Erdbebenherdes. We benutzen die geografische Länge und Breite als Variablen, auf die we projizieren, and die Tiefe als Covariable, nach der we Schnitte bilden.

The Tiefen codieren we um, damit bei grafischen Darstellungen große Tiefen nach unten zeigen.

*Input*

---

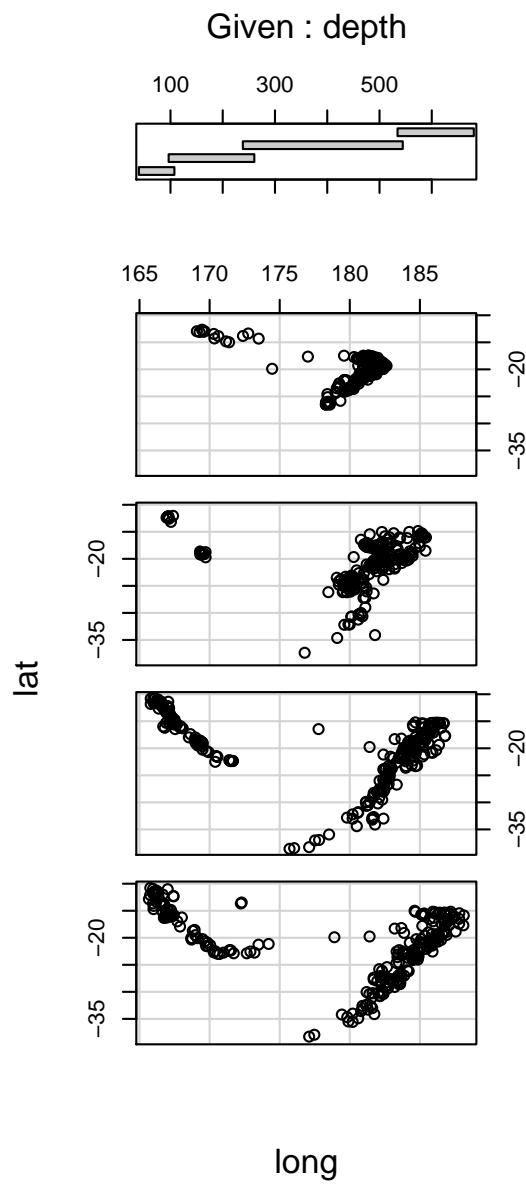
```

quakes$depth <- -quakes$depth
given.depth <- co.intervals(quakes$depth, number = 4, overlap = .1)
coplot(lat ~ long | depth, data = quakes, given.values = given.depth, columns = 1)

```

**ToDo:**

or change  
plot coor-  
dinates?



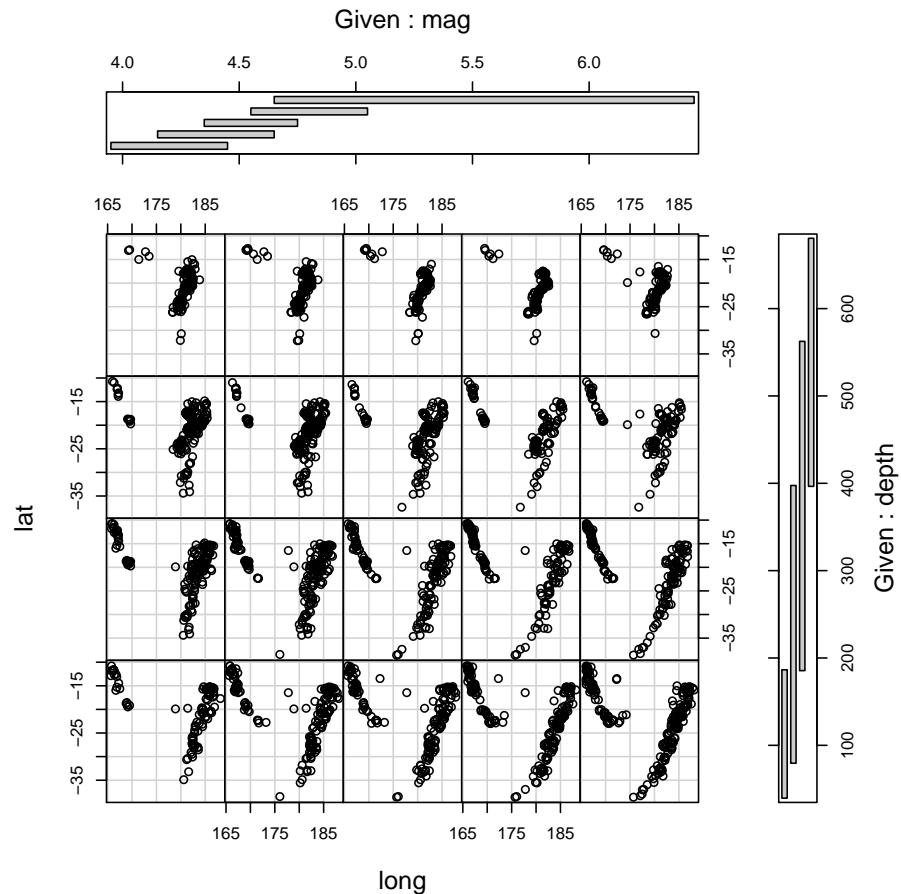
Analog for zwei Covariable, die Tiefe und die Stärke des Erdbebens.

---

*Input*

```
coplot(lat ~ long | mag * depth , data = quakes, number = c(5, 4))
```

---



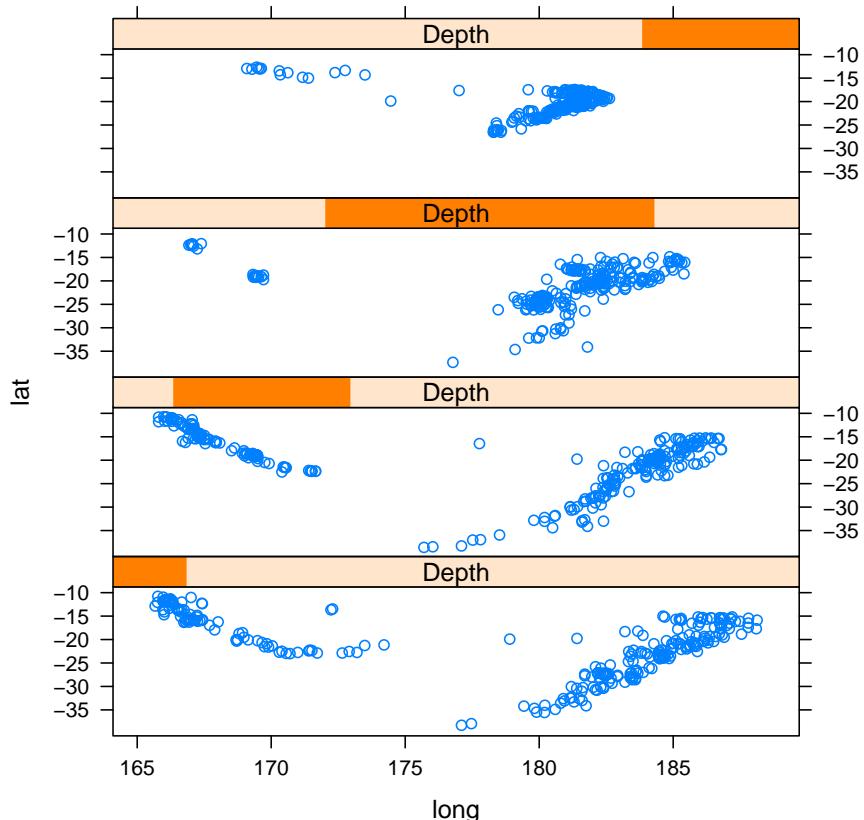
a:ch04:5

Aufgabe 4.4	
	Analysieren Sie den "quakes"-Datensatz. Fassen Sie Ihre Ergebnisse zusammen. Versuchen Sie, ein formales Modell zu formulieren.
	Wie hängt die geographische Position mit der Tiefe zusammen?
	Ist ein Zusammenhang von Tiefe und Stärke des Erdbebens erkennbar? (Evtl. müssen Sie bei <code>coplot()</code> eine andere Formel wählen.)

The Idee der Coplots wird generalisiert in den Trellis-Displays (siehe [4]). Trellis-Displays are in R in `library("lattice")` implementiert. Input [clev93vd](#)

```
library("lattice")
Depth <- equal.count(quakes$depth, number = 4, overlap = .1)
print(xyplot(lat ~ long | Depth , data = quakes, columns = 1, layout = c(1, 4)))
```

**To Do:**  
Comment  
on trellis



**ToDo:**  
add mag

**ToDo:** Ref  
to curse of  
dimension  
**ToDo:** Va-  
riance bias  
dilemma

#### 4.6 Transformations and Dimension Reduktion

Variable liegen oft in der Form vor, die von den Messprozessen or fachlichen conventions vorgegeben are. Sie entspricht nicht unbedingt der Form, die von der Sache her vorgegeben is, or die for die statistische Modellierung am besten geeignet is. Diese Form enthält eine gewisse Beliebigkeit:

- Bei einer akustischen Reizbestimmung kann die Stärke der Reizes zum Beispiel durch die Energie beschrieben werden, or durch den Schalldruck [Phon]. Von der einen zur anderen Skala führt die Logarithmus- Transformation. Das Weber-Fechnersche Gesetz der Psychologie sagt, dass for die menschliche Wahrnehmung die (logarithmische) Phon-Skala die richtige is.
- Benzinverbrauch wird in den USA als Miles per Gallon angegeben, in Europa als Liter auf 100 km. Bis auf eine Umrechnungskonstante is die eine Variable das inverse der anderen. The Angabe in Liter auf 100 km scheint zu einfacheren statistischen Modellen zu führen; Analysen in Miles per Gallon können beliebig kompliziert sein.

The Wahl der richtigen Variablen kann ein entscheidender Schritt in der Analyse sein. Dabei kann es hilfreich sein, zunächst Transformationen und zusätzliche konstruierte Variablen einzuführen, and dann in a zweiten Schritt die Dimension wieder zu reduzieren and die effektiven Variablen zu bestimmen.

`plot@plot|textit`

Koordinatensysteme are nicht kanonisch vorgegeben. Dies trifft schon auf univariate Probleme zu. Bei univariaten Problemen können we Koordinatensysteme noch relativ einfach transformieren. The Modellierung der Fehlerverteilung einerseits and die Transformation der Daten auf eine Standard-distribution are in gewisser Weise austauschbar. In mehrdimensionalen Situationen are geeignete Transformationsfamilien bisweilen nicht verfügbar or nicht zugänglich, and die Struktur des Problems kann kritisch von der Wahl geeigneter Koordinaten abhängig sein. Hier hat eine sachorientierte Wahl der Koordinatendarstellung oft den Vorzug vor automatischen Selektionen.

Dieses kann an Anderson's Iris-Datensatz illustriert werden. Der Datensatz hat fünf Dimensionen: vier quantitative Variable (Länge and Breite von Blütenblatt (engl. petal) and Kelchblatt (engl. sepal) von Iris-Blüten) and eine kategoriale Variable (die Spezies: iris setosa canadensis, iris versicolor, iris virginica)\*. Gesucht is eine Klassifikation der Spezies anhand der vier quantitativen Variablen.

Table 4.11 *Iris Species.*



The Struktur is ähnlich der des Diabetes-Datensatzes `chemdiab`. The Klassifikation nach `iris$Species` is hier jedoch eine (extern) gegebene Klassifikation, im Gegensatz zur anhand der anderen Variablen definierten Klassifikation `chemdiab$cc`. Gesucht is hier nicht eine allgemeine Beschreibung wie bei `chemdiab`, sondern eine Klassifikationsregel, die `iris$Species` aus den anderen Variablen ableitet.

The Spezies definieren die Selektionen, die in diesem Beispiel von Interesse are.

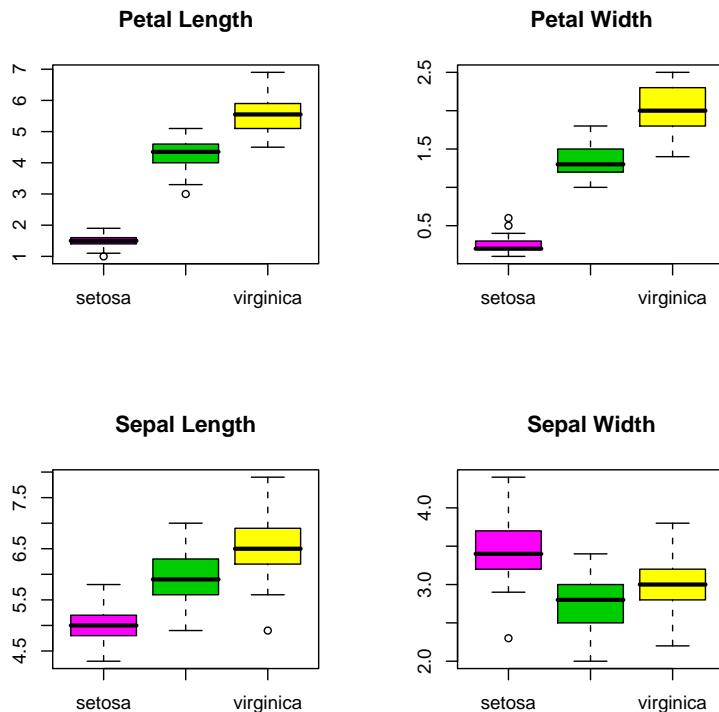
Um eine erste Übersicht zu bekommen is es naheliegend, die vier Variablen getrennt nach Spezies zu betrachten. The Standard-conventions von R machen dies umständlich. The Spezies is eine kategoriale Variable. Dies veranlasst R, bei der `plot()`-function von einer Punkt-Darstellung zu Box&Whisker-Plots überzugehen.

*Input*

```
oldpar <- par(mfrow = c(2, 2))
plot(iris$Species, iris$Petal.Length,
     ylab = '', main = 'Petal Length', col = c("magenta", "green3", "yellow"))
```

\* Photos: The Species Iris Group of North America. Mit freundlicher Genehmigung

```
stripplot@stripplot{textit,
  plot(iris$Species, iris$Petal.Width,
       ylab = '', main = 'Petal Width', col = c("magenta", "green3", "yellow"))
  plot(iris$Species, iris$Sepal.Length,
       ylab = '', main = 'Sepal Length', col = c("magenta", "green3", "yellow"))
  plot(iris$Species, iris$Sepal.Width,
       ylab = '', main = 'Sepal Width', col = c("magenta", "green3", "yellow"))
  par(oldpar)}
```



We könnten die R-Functions modifizieren, um einen Scatterplot der einzelnen Variablen nach Gruppen zu erhalten. Anstelle dessen greifen we wieder auf `grid` and `lattice` zurück und benutzen die function `stripplot()`. Weil bei der gegebenen Messgenauigkeit values vielfach auftreten, benutzen we ein 'jitter': we 'verwackeln' vielfache values, um sie getrennt darzustellen.

---

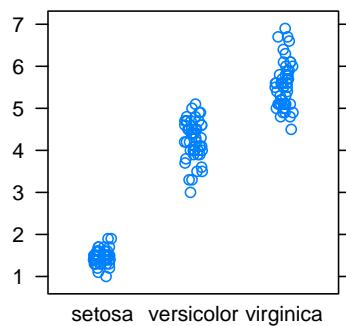
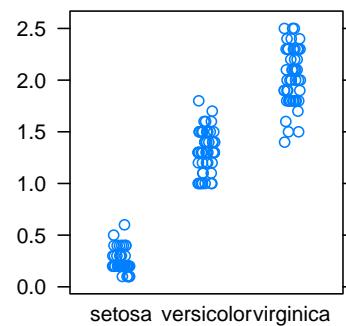
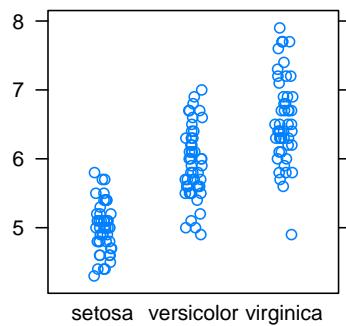
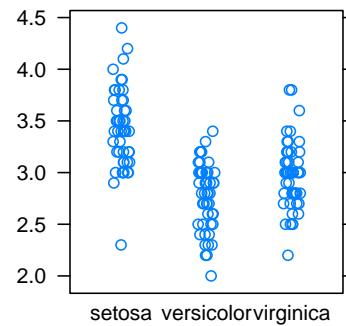
*Input*

---

```
library("lattice")
print(stripplot(Petal.Length ~ Species, data = iris,
               jitter = TRUE, ylab = '', main = 'Petal Length'), split = c(1, 1, 2, 2), more = TRUE)
print(stripplot(Petal.Width ~ Species, data = iris,
               jitter = TRUE, ylab = '', main = 'Petal Width'), split = c(2, 1, 2, 2), more = TRUE)
print(stripplot(Sepal.Length ~ Species, data = iris,
               jitter = TRUE, ylab = '', main = 'Sepal Length'), split = c(1, 2, 2, 2), more = TRUE)
```

```
print(stripplot(Sepal.Width ~ Species, data = iris,
                jitter = TRUE, ylab = '', main = 'Sepal Width'), split = c(2, 2, 2, 2))
```

lda@lda|textit

**Petal Length****Petal Width****Sepal Length****Sepal Width**

The eindimensionalen Randverteilungen geben noch wenig Hinweis darauf, wie die drei Gruppen zu trennen sind. Auch die zweidimensionale Darstellung hilft wenig weiter.

a:ch04:11

**To Do:**  
add color

**To Do:**  
Remark  
n small  
setosa

**Aufgabe 4.5**

Benutzen Sie die Methoden aus Abschnitt 4.4 und 4.5, um den Datensatz ch04projektionen zu untersuchen. Können Sie Klassifikationsregeln erkennen, die die drei Spezies weitgehend richtig klassifizieren?

Mit formalen Methoden wie der Diskriminanzanalyse (z. B. `lda()` in `library (MASS)`) kann die Klassifikation anhand der ursprünglichen Variablen gefunden werden. Die Trennung der Spezies ist nicht trivial.

The ursprünglichen Variablen repräsentieren jedoch nur den Aspekt der Daten, der technisch am einfachsten erhebbar ist. Biologisch gesehen würde man jedoch anders parametrisieren: die Variablen spiegeln Größe und Form der Blätter wieder. Eine erste Approximation wäre

$$area = length \cdot width \quad (4.1)$$

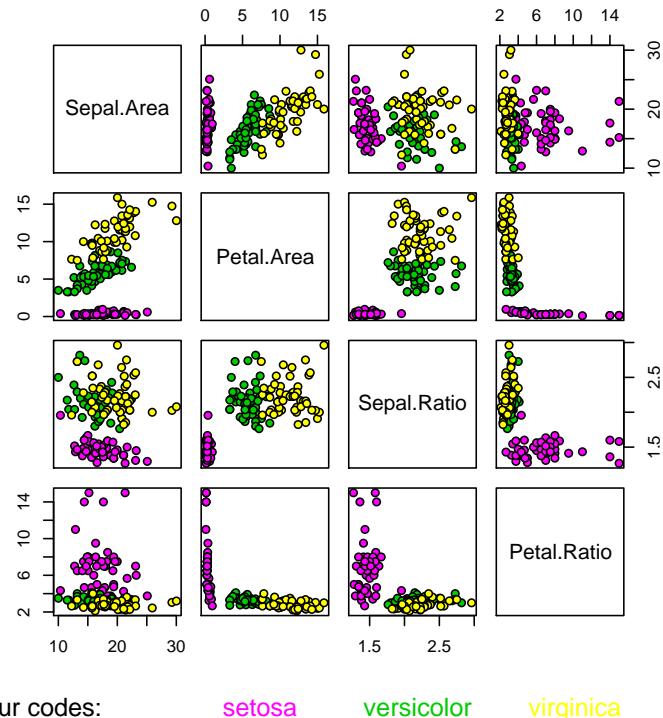
$$aspectratio = length / width. \quad (4.2)$$

**ToDo:**

check iris  
inverse  
ratio

Damit erhält man die Darstellung *Input*  
`iris$Sepal.Area <- iris$Sepal.Length * iris$Sepal.Width`  
`iris$Petal.Area <- iris$Petal.Length * iris$Petal.Width`  
`iris$Sepal.Ratio <- iris$Sepal.Length / iris$Sepal.Width`  
`iris$Petal.Ratio <- iris$Petal.Length / iris$Petal.Width`  
`pairs(iris[6:9], main = "Anderson's Iris Data -- 3 species",`  
`pch = 21,`  
`bg = c("magenta", "green3", "yellow") [unclass(iris$Species)],`  
`oma = c(8, 8, 8, 8))`  
`mtext(c("Colour codes:", levels(iris$Species)),`  
`col = c("black", "magenta", "green3", "yellow"),`  
`at = c(0.1, 0.4, 0.6, 0.8),`  
`side = 1, line = 2)`

**Anderson's Iris Data -- 3 species**



Colour codes:

setosa

versicolor

virginica

**ToDo:** use  
attach

In der Marginalverteilung are die Spezies fast vollständig getrennt - with zwei Grenzfällen. In diesen mehr biologischen Koordinaten sieht man, dass zur Klassifikation Fläche and Längenverhältnis des Blütenblatts allein ausreichen. Jedes kompliziertere formale Verfahren muss sich with dieser trivialen Klassifikationsregel erst einmal messen.

Selbst eine umfassende Suche, z.B. with projection pursuit, erfasst nur die Projektionen, also nur spezielle Linearkombinationen der Variablen. Bei den Iris-Daten haben we zunächst neue Variablen, die Flächen and Seitenverhältnisse, eingeführt. Dies are nichtlineare Transformationen. Erst in a weiteren Schritt are dann die klassifizierenden Variablen identifiziert worden, and dabei is die Dimension drastisch reduziert. Bei echten multivariaten Problemen is es ganz typisch, dass zunächst eine Dimensionserweiterung notwendig is, um das Problem zu lösen. Dimensionsreduktion is erst dann sinnvoll, wenn die beschreibenden Variablen hinreichend komplex are, um zu einer Lösung zu führen.

## 4.7 Higher Dimensions

### 4.7.1 Linear Case

Haben we im wesentlichen lineare Strukturen, so können we oft auch höher-dimensionale Strukturen with Methoden analysieren, die for eindimensionale Modelle entwickelt are. We müssen die Methoden evtl. modifizieren or iteriert anwenden. Sie helfen uns jedoch, die wesentlichen Merkmale zu erkennen.

In Kapitel 2 haben we lineare Modelle bereits allgemein for beliebige Dimension  $p$  der Regressoren eingeführt and damit bereits den mehrdimensionalen Fall eingeschlossen. Kapitel 2 setzt voraus, dass das Modell der statistischen Analyse vorab fest steht, d.h. das Information aus dem Datenmaterial die Wahl des Modells nicht beeinflusst, sondern nur die Entscheidung im Rahmen des Modells.

Insbesondere bei höherdimensionalen Problemen is es jedoch so, dass das Modell erst zu bestimmen is. Ein wichtiger Spezialfall is die Auswahl von Regressoren: die Variablen are Kandidaten, aus denen eine (möglichst kleine) Anzahl von Regressoren zu wählen is.

Bringen kompliziertere Modelle eine wesentliche Verbesserung gegenüber dem einfachen Modell? Welche parameter resp. welche abgeleitete Variable sollten in das Modell einbezogen werden? The Lehre aus den linearen Modellen is, dass nicht der value des einzelnen Parameters den Beitrag im Modell bestimmt, sondern dass die durch die parameter bestimmten Räume die wesentlichen Faktoren are. An dieser Stelle are angepasste Strategien gefragt. We können with einfachen Modellen begin and fragen, ob zusätzliche parameter einen weiteren Beitrag liefern. Dadurch erreichen we einen besseren Fit, aber erhöhen die Varianz unserer Schätzungen. or we können with a relativ komplexen Modell begin, and fragen, ob we parameter fortlassen können. Dadurch wird zwar der Restfehler erhöht, we gewinnen aber an Verlässlichkeit der Schätzungen.

Beide Strategien führen im abstrakten linearen Regressionsmodell zu a Vergleich von zwei Modellräumen  $\mathcal{M}_{X'} \subset \mathcal{M}_X$ . The entsprechenden Schätzer are  $\pi_{\mathcal{M}_{X'}}(Y)$  and  $\pi_{\mathcal{M}_X}(Y)$ . The Beziehung zwischen beiden wird klar, wenn we die orthogonale Zerlegung  $\mathcal{M}_X = \mathcal{M}_{X'} \oplus L_X := M_0, L_X := \mathcal{M}_X \ominus \mathcal{M}_{X'}$  von  $\mathcal{M}_X$  wählen. Dann is  $\pi_{\mathcal{M}_X}(Y) = \pi_{\mathcal{M}_{X'}}(Y) + \pi_{L_X}(Y)$ .

**ToDo:**  
formal  
variance  
bias

**Added-Variable-Plots** | **Partial Residuals and Added Variable Plots**

In der Regression are  $\mathcal{M}_{X'}$  and  $\mathcal{M}_X$  Räume, die von den Regressor-Variablenvektoren aufgespannt werden. In unserer Situation interessiert uns der Spezialfall

$$X' = \text{span}(X_1, \dots, X_{p'}); X = \text{span}(X_1, \dots, X_p)$$

with  $p > p'$ . Dann wird aber  $L_X$  aufgespannt von den vectors

$$R_{p'+1} = X_{p'+1} - \pi_{\mathcal{M}'_X}(X_{p'+1}), \dots, R_p = X_p - \pi_{\mathcal{M}'_X}(X_p).$$

Wenn we also (formal) eine lineare Regression der zusätzlichen Regressoren nach den bereits in  $X'$  enthaltenen durchführen, are die dabei entstehenden Residuen ein Erzeugendensystem für  $L_X$ . Eine weitere Regression von  $Y$  nach diesen Residuen liefert uns den Term  $\pi_{L_{X'}}(Y)$ , der den Unterschied zwischen den Modellen beschreibt. Nach Konstruktion wissen we, dass  $\pi_{\mathcal{M}'_X}(Y)$  orthogonal zu  $L_X$  is. Bei dieser zweiten Regression wird deshalb dieser Anteil auf null abgebildet. We können diesen Anteil gleich eliminieren and uns auf die Regression von  $Y' = Y - \pi_{\mathcal{M}'_X}(Y)$  nach  $R_{p'+1}, \dots, R_p$  beschränken.

The Strategiewahl is einfach: we untersuchen, ob zusätzliche parameter in das Modell aufgenommen werden sollten. Anstelle der Scatterplot-Matrix der ursprünglichen Daten betrachten we die Scatterplots der (formalen) Residuen aus diesem einfachen Modell. Diese Scatterplots werden **Added-Variable-Plots** genannt.

Um den Unterschied zur Scatterplot-Matrix der Ausgangsdaten zu betonen: lineare Strukturen im Scatterplot der Ausgangsdaten are ein klarer Hinweis auf lineare Abhängigkeiten. Nicht-lineare Strukturen, wie z.B. die Dreiecksgestalt in einigen der Scatterplots können eine entsprechende Abhängigkeit widerspiegeln; sie können aber auch Artefakte sein, die als sequence der Verteilungs- und Korrelationsstruktur der Regressoren auftreten. Sie haben in der Regel keine einfache Deutung. Im Gegensatz dazu are die Darstellungen in der Matrix der Added-Variable-Plots for lineare Effekte der vorausgehenden Variablen adjustiert. Dadurch hängen sie von der Wahl der Reihenfolge ab, in der Variable einbezogen werden. Sie korrigieren aber for lineare Effekte, die aus den Korrelationen zu vorausgehenden Variablen kommen. Dadurch wird eine ganze Reihe von Artefakten vermieden and sie können unter Berücksichtigung des Zusammenhangs unmittelbar interpretiert werden.

a:ch04:12

Aufgabe 4.6	
	<p>Modifizieren Sie die nachfolgende Prozedur <code>pairslm()</code> so, dass sie for alle Variablen in der ursprünglichen Matrix <code>x</code> die Residuen der Regression nach der neuen Variablen <code>x\$fit</code> berechnet and eine Scatterplot-Matrix dieser Residuen zeigt.</p> <pre><code>pairslm &lt;- function(model, x, ... )   { x\$fit &lt;- lm(model, x)\$fit; pairs(x, ...)}</code></pre> <p>Fügen Sie auch Titel, Legenden etc. hinzu.</p> <p>Benutzen Sie den "trees"-Datensatz als Beispiel.</p>

We haben den Übergang von  $p'$  zu  $p' + 1$  Variablen untersucht. The Scatterplot-Matrix erlaubt uns einen schnellen Überblick über eine (nicht zu) große Zahl von Kandidaten (bei uns drei mögliche zusätzliche Regressoren). Der Übergang von  $p$  zu  $p - 1$ , zur Elimination einer Varia-

```
prcomp@prcomp|textit
mva@mva|textit
```

blen, is in gewisser Weise dual dazu. Dies entspricht der zweiten Strategie, der schrittweisen Elimination.

Statt eine einzelne Variable als Leitvariable auszuwählen ist es effizienter, Kombinationen von Variablen als synthetische Leitvariablen zu benutzen. Entsprechende Methoden werden in der Theorie als Hauptkomponentenanalyse behandelt und durch die function `prcomp()` in der `library(mva)` bereitgestellt. We kommen daraus in a späteren Beispiel (Seite 4-42) zurück.

Das Beispiel der linearen Modelle lehrt uns, dass die marginalen Beziehungen nur die halbe Wahrheit are. Anstelle die einzelnen Regressoren zu betrachten, müssen we im linearen Modell schrittweise orthogonalisieren. Komponentenweise Interpretationen are damit fragwürdig - sie are weitgehend von der Reihenfolge abhängig, in der Variablen einbezogen werden.

In komplexeren Situationen führen formale Methoden oft nur in die Irre. Handwerkliches Geschick is hier notwendig. Leider are die Kennnntnisse darüber, wie handwerkliche Eingriffe die Gültigkeit formaler Methoden beeinflussen, noch sehr beschränkt. Deshalb is es gerade hier wichtig, gewählte Strategien anhand von Simulationen kritisch zu beurteilen.

#### 4.7.2 Nonlinear Case

Nichtlineare Beziehungen in höheren Dimension stellen eine Herausforderung dar. Neben den Methoden brauchen we auch ein Repertoire an examplesn, die uns zeigen, welche Strukturen auftreten können and worauf we achten müssen. Das folgende Beispiel, Cusp(Spitzen)-Singularität gehört dazu: es is with die einfachste Struktur, die in höheren Dimensionen auftreten kann. The Basis is hier ein zweidimensionale Struktur, eine Fläche, die nicht trivial in a dreidimensionale Raum eingebettet is. Das interessante Merkmal is hier die Aufspaltung von einer unimodalen in eine bimodale Situation.

##### *Example: Cusp Nonlinearity*

Das einfachste Beispiel kann im Hinblick auf physikalische Anwendungen illustriert werden. In physikalischen Systemen hängen probabilitätsverteilungen oft with Energiezuständen zusammen; (lokale) Minima der Energie entsprechen dabei den Moden der distribution. Ein typischer Zusammenhang is: verhält sich die Energie wie  $\varphi(y)$ , so verhält sich die distribution nach Standardisierung wie  $e^{-\varphi(y)}$ . Ist  $\varphi(y)$  in der Nähe des Minimums quadratisch, so erhalten we (bis auf Skalentransformation) distributions aus der Familie der Normalverteilungen.

The Differentialtopologie lehrt uns, dass auch bei kleinen Störungen or Variationen dieses qualitative Bild erhalten bleibt. The Energie bleibt zumindest lokal approximativ quadratisch, and die Normalverteilungen bleiben zumindest approximativ eine geeignete Verteilungsfamilie.

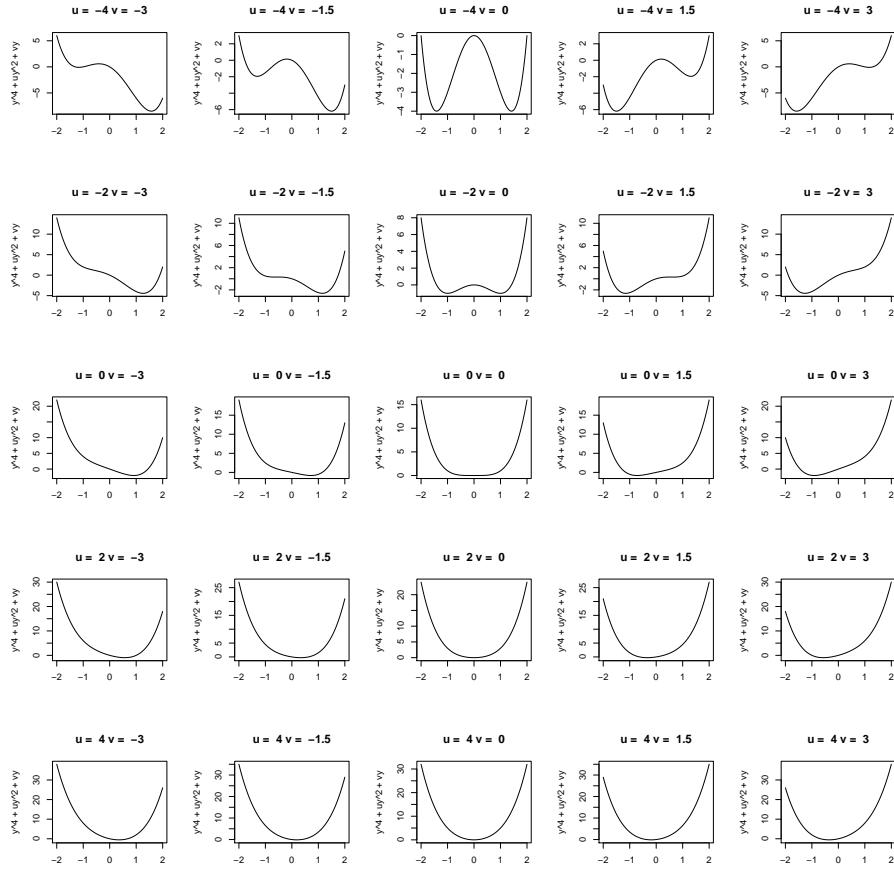
Das Verhalten ändert sich drastisch, wenn das Potential sich lokal wie  $y^4$  verhält. Schon geringe Variationen können dazu führen, dass das Potential lokal quadratisch is. Aber sie können auch dazu führen, dass das lokale Minimum aufbricht and zu zwei Minima führt. Das typische Bild is von der Gestalt

$$\varphi(y; u, v) = y^4 + u \cdot y^2 + v \cdot y. \quad (4.3)$$

f:cusp0

Dabei are die Variationen durch die parameter  $u, v$  repräsentiert. Am einfachsten lässt sich die Situation dynamisch interpretieren: we stellen uns vor, dass  $u, v$  äußere parameter are,

4-30

DIMENSIONS 1, 2, 3, ...,  $\infty$ Figure 4.1 *Entfaltung von  $y^4$ :  $\varphi(y; u, v) = y^4 + u \cdot y^2 + v \cdot y$* 

die sich verändern können. Dieses Bild kennen we von der magnetischen Hysterese:  $y$  gibt die Magnetisierung in einer Richtung an,  $u$  spielt die Rolle der Temperatur;  $v$  die eines äußeren Magnetfelds. Bei hoher Temperatur folgt die Magnetisierung direkt dem äußeren Magnetfeld. Sinkt die Temperatur, so zeigt das Material Gedächtnis: die Magnetisierung hängt nicht nur vom äußeren Magnetfeld ab, sondern auch von der vorhergehenden Magnetisierung.

Ähnliche "Gedächtniseffekte" kennen we auch in anderen Bereichen. Man stelle sich einen Markt vor with Preisen  $y$ , Kosten  $v$  and a "Konkurrenzdruck"  $u$ . Bei ausreichender Konkurrenz folgen die Preise (mehr or weniger) den Kosten bei sonst gleichen Bedingungen. Bei Monopol-Situationen scheinen die Preise ein Gedächtnis zu haben: are sie einmal gestiegen, so sinken sie erst, wenn die Kosten drastisch reduziert are.

The in Formel 4.3 angegebenen "Entfaltung" des Potentials  $y^4$  hat eine typische Form. Aus

$$\varphi'(y; u, v) = 4y^3 + 2u \cdot y + v = 0 \quad (4.4)$$

erhält man die kritischen Punkte (siehe Abb. 4.2). Siehe Abbildung 4.2 auf Seite 4-31.

f:cusp0

f:cusp1

fig:ch04cuspssurf1

fig:f04cuspssurf1

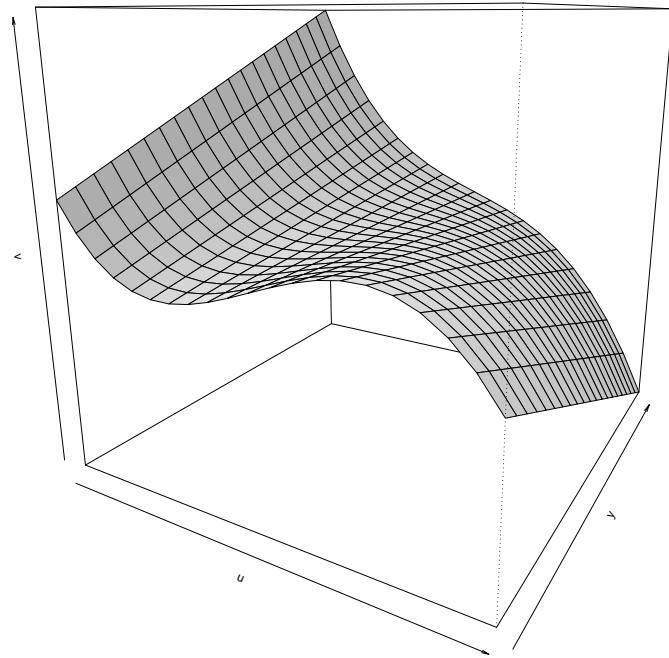


Figure 4.2 Kritische Punkte  $\varphi'(y; u, v) = 4y^3 + 2u \cdot y + v = 0$

`fig:ch04cuspsurf1`

Projiziert auf die  $u, v$ -Ebene gibt dies eine Spitze (engl.: “cusp”, Abb. 4.3). Bei Parametern im inneren dieser Spitze gibt es zwei lokale Minima; außerhalb der Spitze gibt es nur einen Extremalwert.

**ToDo:** adjust scale, orientation

`ch04cuspline`

The diesen Potentialen entsprechenden distributions are – bis auf Skalentransformation zur Normalisierung –

$$p(y; u, v) \propto e^{-(y^4 + u \cdot y^2 + v \cdot y)}. \quad (4.5)$$

**ToDo:** 2/3

The Struktur der Potentiale spiegelt sich auch in den entsprechenden distributions wieder; der exponentielle Abfall macht allerdings die kritische Grenze etwas komplizierter.

`propy4exp`

The Situation erscheint hier noch harmlos: der Parameterraum (der Raum der Regressoren  $x = (u, v)$ ) hat nur zwei Dimensionen. The distribution is eindimensional with einer glatten Dichte. Aber die Situation kann with linearen Methoden nur unzureichend erfasst werden. Der

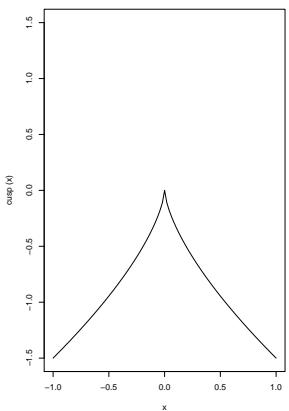


Figure 4.3 Grenze zwischen Uni- und Bimodalität im  $(u, v)$ -Raum

typische nichtlineare Effekt wird nicht erkannt, wenn man darauf nicht vorbereitet ist. Erst das Gesamtbild im Dreidimensionalen vermittelt die eigentliche Struktur.

Dieses einfache Beispiel ist eine Herausforderung. Wie kann eine derartige Struktur diagnostiziert werden?

a:ch04:13

Aufgabe 4.7	
	Schreiben Sie eine function <code>dx4exp(x, u, v)</code> , die die zentrierte probabilitätsdichte zu (4.5) berechnet. Dazu müssen Sie die Dichte aus (4.5) integrieren, um die Normierungskonstante zu bestimmen, und den Erwartungswert berechnen, um die Dichte zu zentrieren. Benutzen Sie for beides eine numerische Integration with <code>integrate()</code> .
***	Simulieren Sie valuesn $u, v$ auf a Gitter in $u = -2 \dots 2$ and $v = -1 \dots 1$ je 100 random numbers aus <code>dx4exp(x, u, v)</code> . Untersuchen Sie diese with den Methoden aus Kapitel 2. Können Sie Hinweise auf nicht-lineare Abhängigkeit erkennen? Ist die Bimodalität erkennbar? Wie weit können Sie die Struktur identifizieren?

Bei nichtlinearen Beziehungen können gemeinsame Abhängigkeiten eine große Bedeutung haben. Im allgemeinen erfordert dies Umsicht bei der Modellbildung. Nichtlineare Beziehungen können in Projektionen versteckt sein. Artefakte der (linearen) Projektion können ein Bild vermitteln, das nicht den ursprünglichen Beziehungen entspricht.

## HIGHER DIMENSIONS

4-33

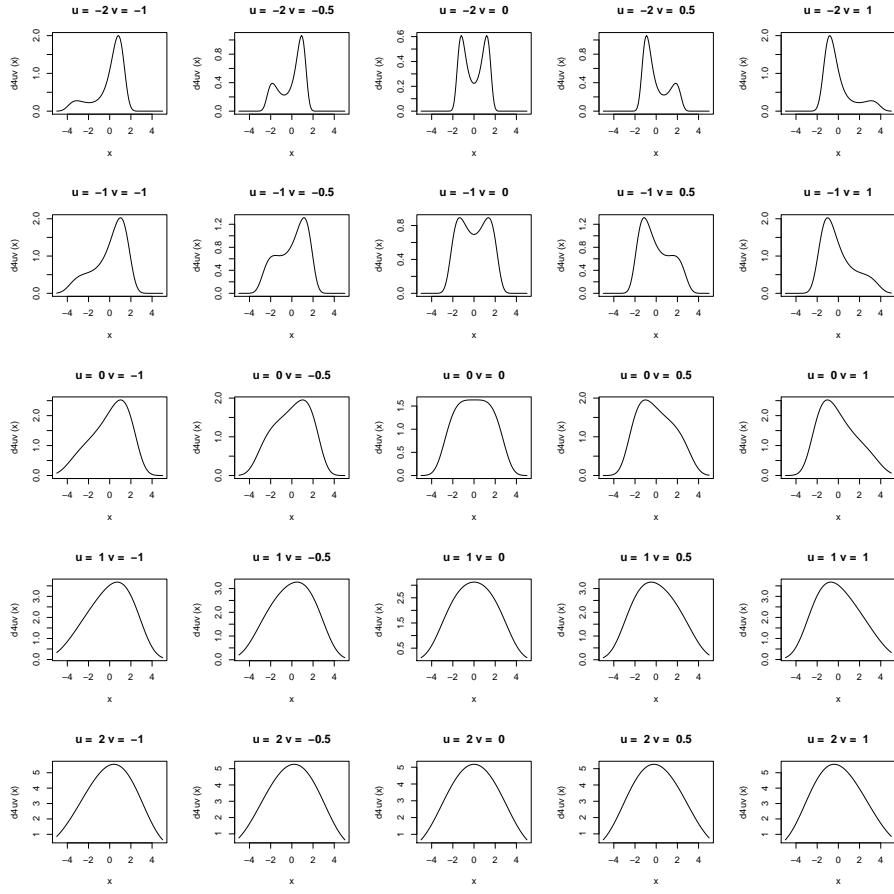


Figure 4.4  $p(y; u, v) \propto e^{-(y^4 + u \cdot y^2 + v \cdot y)}$

### ~~curse of dimension~~ $\text{4.7.3}$ “Curse of Dimension”

Ohne angepasste Koordinatensysteme ist eine umfassende Suche nach interessanten Projektionen und Schnitten nötig. The Anzahl der Möglichkeiten steigt rasch with der Dimension. Zur Illustration: Um einen Kubus zu identifizieren müssen zumindest die Eckpunkte erkannt werden. In  $d$  Dimensionen are dies  $2^d$  Eckpunkte. The Anzahl steigt exponentiell with der Dimension. Dies is ein Aspekt des als **curse of dimension** bekannten Problems.

Anders betrachtet: bezeichnen we die Datenpunkte, die in mindestens einer Variablen dimension extrem are, so are dies im eindimensionalen Fall zwei Punkte. in  $d$  Dimensionen are dies typischerweise  $2^d$  Punkte. Betrachten we nicht nur Koordinatenrichtungen, sondern beliebige Richtungen, so is typisch jeder Punkt extremal, wenn  $d$  sehr groß wird.

Ein dritter Aspekt: im  $d$ -dimensionalen Raum is fast jeder Punkt isoliert. Lokalisierungen, wie we sie in Abschnitt  $\text{2.5}$  kennengelernt haben, brechen zusammen. Wählen we um einen Punkt eine Umgebung, die einen Anteil  $p$ , z.B.  $p = 10\%$  der Variablenspannweite umfasst, so haben we in einer Dimension typischerweise der Größenordnung nach auch einen Anteil  $p$  der Datenpunkte erfasst. In  $d$  Dimensionen is dies nur noch ein Anteil der Größenordnung  $p^d$ . Bei zum Beispiel 6 Dimensionen brauchen we also mehrere Millionen Datenpunkte, damit we nicht with leeren Umgebungen arbeiten.

#### 4.7.4 Case Study: Body Fat

Als fortlaufendes Beispiel benutzen we nun den Fat-Datensatz. Dieser Datensatz is in der Literatur wiederholt veröffentlicht and in R unter anderem im Paket *UsingR* zugänglich.

Ziel der Untersuchung hinter diesem Datensatz is die Bestimmung des Körperfettanteils. The verlässlichste Methode is es, in a Wasserbad die mittlere Dichte des Gewebes zu bestimmen and daraus auf den Körperfettanteil zurück zu schliessen. Diese Bestimmung is sehr aufwendig. Kann sich durch einfacher zu messende Körperparameter ersetzt werden? The zur Verfügung stehenden parameter are in Tabelle  $\text{4.15}$  zusammengefasst.

Anhand der Übersicht in Tabelle  $\text{4.15}$  sehen we gleich, dass metrische Angaben and US-Maße gemischt are. Damit for uns die Interpretation einfacher is, stellen we alle Angaben auf metrische values um.

---

Input
library("UsingR")
data(fat)
fat\$weightkg <- fat\$weight*0.453
fat\$heightcm <- fat\$height * 2.54
fat\$ffweightkg <- fat\$ffweight*0.453

---

The Variablen *body.fat* and *body.fat.siri* are aus dem gemessenen value *density* abgeleitet. Hinter den Formeln stecken Annahmen über die mittlere Dichte von Fett and von fettfreiem Gewebe. Mit diesen Annahmen kann aus *density* der Fettanteil errechnet (or besser: geschätzt) werden. In beiden Formeln is der dichtabhängige Faktor  $1/density$ . Bis auf (gegebene or angenommene) Konstanten is dies also der for uns relevante Term (and nicht *density*).

Der erste Schritt is eine kritische Inspektion and Bereinigung des Datensatzes. Dies is fast immer nötig, nicht nur bei höherdimensionalen Datensätzen. Bei höherdimensionalen Datensätzen

`pairs@pairs|textit`

name	Variable	Einheit, Bem.
case	Case Number	
body.fat	Percent body fat using Brozek's equation, $457/Density - 414.2$	
body.fat.siri	Percent body fat using Siri's equation, $495/Density - 450$	
density	Density	[g/cm <sup>2</sup> ]
age	Age	[yrs]
weight	Weight	[lbs]
height	Height	[inches]
BMI	Adiposity index = Weight/Height <sup>2</sup>	[kg/m <sup>2</sup> ]
ffweight	Fat Free Weight = (1 - fraction of body fat) * Weight, using Brozek's formula	[lbs]
neck	Neck circumference	[cm]
chest	Chest circumference	[cm]
abdomen	Abdomen circumference "at the umbilicus and level with the iliac crest"	[cm]
hip	Hip circumference	[cm]
thigh	Thigh circumference	[cm]
knee	Knee circumference	[cm]
ankle	Ankle circumference	[cm]
bicep	Extended biceps circumference	[cm]
forearm	Forearm circumference	[cm]
wrist	Wrist circumference "distal to the styloid processes"	[cm]

Table 4.15 *Fat data set: variables*`fatvars`

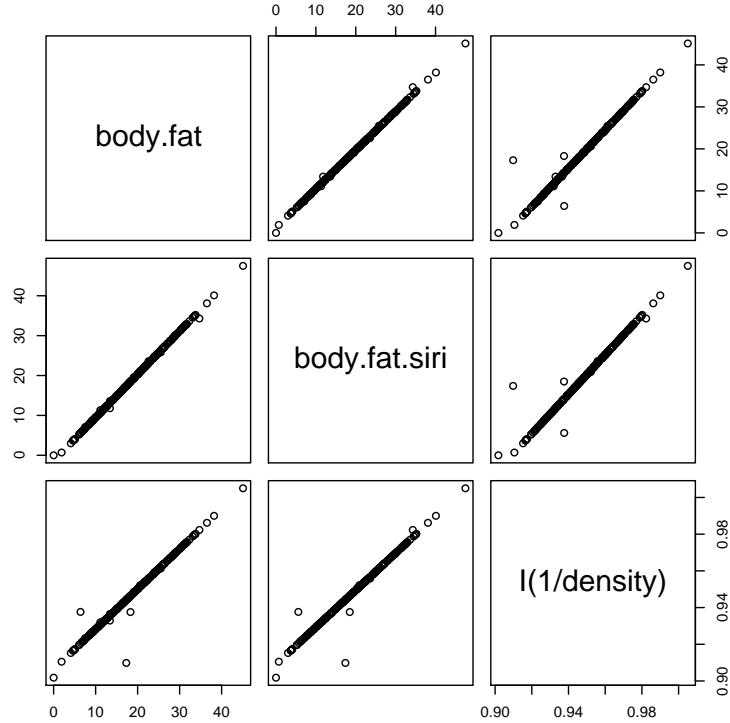
haben we allerdings oft Redundanzen, die Konsistenzprüfungen und evtl. Korrekturen ermöglichen. In unserem Fall are `body.fat`, `body.fat.siri`, `ffweight` and `BMI` abgeleitete Größen, die zu anderen Variablen in deterministischer Beziehung stehen.

We betrachten zunächst die Gruppe `body.fat`, `body.fat.siri`, `1/density`. The paarweisen Scatterplots sollten Geraden zeigen. `pairs()` leistet gute Dienste. We benutzen es hier in der Formel-Variante. Um zu signalisieren, dass `1/density` berechnet werden soll, and die Division nicht als Formel-Operator zu verstehen is, markieren we den Term entsprechend.

---

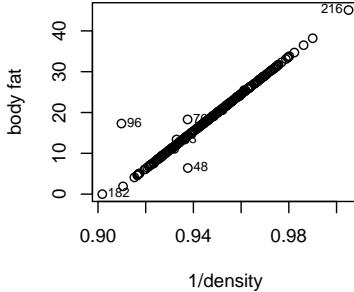
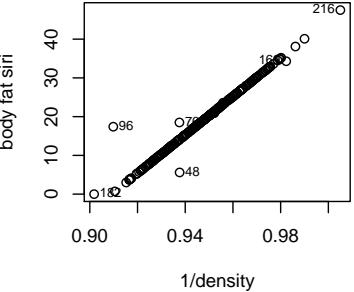
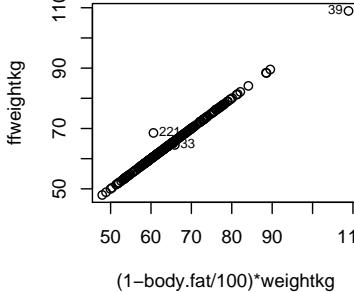
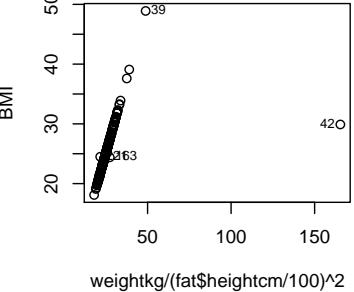
*Input*  
`pairs(~body.fat + body.fat.siri + I(1/density), data = fat)`

---



The inkonsistenten values and Ausreißer are deutlich. Leider ist es in R nicht einfach möglich, values in der Scatterplot-Matrix zu markieren.

a:ch04:14

Aufgabe 4.8	Benutzen Sie <code>plot()</code> and <code>identify()</code> , um die folgenden Ausgaben zu erzeugen:
	
	
	
	

Wenn eine klare Korrektur vorgenommen werden kann, so sollte es hier getan und im Auswertungsbericht notiert werden. Fall 42 ist einfach: eine Größe von 0.73m bei a Gewicht von 63.5kg is unplausibel and inkonsistent zu BMI 29.9. Aus dem BMI lässt sich die Größe rückrechnen. Der eingetragene value von 29.5 Zoll sollte wohl 69.5 Zoll sein.

---

*Input*

---

```
fat$height [42] <- 69.5
fat$heightcm[42] <- fat$height[42] * 2.54
```

Fall 216 is eine Ermessenssache. The Dichte is extrem niedrig, der BMI extrem hoch. Andererseits passen die Körpermaße zu diesen Extremen. Diese Fall kann ein Ausreisser sein, der die Auswertungen verzerren kann. Es kann aber auch eine besonders informative Beobachtung sein. We notieren ihn als Besonderheit.

Nach dieser Voruntersuchung bereinigen we den Datensatz. The Variablen, die keine Informationen mehr enthalten or die we ersetzt haben, löschen we. Als Zielvariable benutzen we `body.fat`. We behalten jedoch noch die Variable `density` for spätere Zwecke.

---

```
lm@lm|textit
```

---

```
fat$weight <- NULL
fat$height <- NULL
fat$ffweight <- NULL
fat$ffweightkg <- NULL
fat$body.fat.siri <- NULL
```

*Input*

Es gibt eine Reihe von gängigen Indizes (siehe Abb. 4.5). Früher war die Faustformel ‘Idealgewicht = Körpergröße -100’ gängig. Heute ist der “body mass index” BMI = Gewicht/Körpergröße<sup>2</sup> gängig. (Handelsübliche Körperfettwaagen bestimmen die elektrische Impedanz. Diese Variable ist im Fat-Datensatz nicht enthalten.)

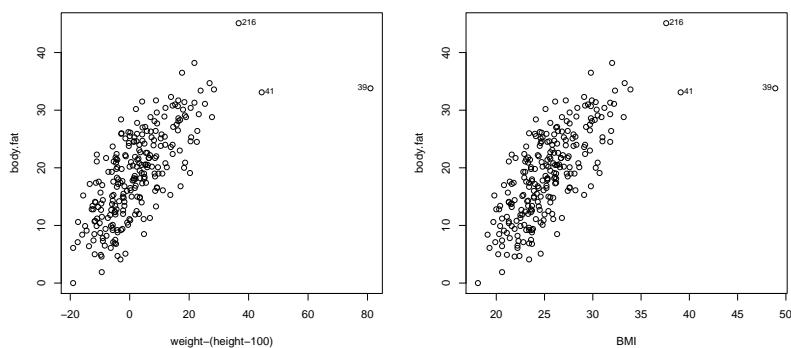


Figure 4.5 Fettanteil gegen konventionelle Indizes

**fat01scat**

We können mit den konventionellen Indizes im linearen Modell schätzen. Dabei lassen wir die offensichtlichen Ausreißer und möglichen Hebelpunkte unberücksichtigt. Dazu benutzen wir den *subset*-parameter der function *lm()*.

für die Faustformel ‘Idealgewicht = Körpergröße -100’ erhalten wir:

---

```
lm.height <- lm(body.fat~I(weightkg-(heightcm-100)),
  data = fat,
  subset = -c(39, 41, 216))
summary(lm.height)
```

*Output*

Call:

```
lm(formula = body.fat ~ I(weightkg - (heightcm - 100)), data = fat,
  subset = -c(39, 41, 216))
```

Residuals:

Min	1Q	Median	3Q	Max
-11.90734	-3.68697	-0.05303	3.65458	12.28000

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	17.70722	0.33296	53.18	<2e-16 ***
I(weightkg - (heightcm - 100))	0.54557	0.03283	16.62	<2e-16 ***
---				
Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

Residual standard error: 5.166 on 247 degrees of freedom  
 Multiple R-squared: 0.5279, Adjusted R-squared: 0.526  
 F-statistic: 276.2 on 1 and 247 DF, p-value: < 2.2e-16

The Regression von *body.fat* nach *BMI* results in:

---

Input

```
lm.BMI <- lm(body.fat ~ BMI,
  data = fat,
  subset = -c(39, 41, 216))
summary(lm.BMI)
```

---

Output

Call:

```
lm(formula = body.fat ~ BMI, data = fat, subset = -c(39, 41,
  216))
```

Residuals:

Min	1Q	Median	3Q	Max
-12.49460	-3.53561	-0.05228	3.69129	11.72720

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-25.6130	2.6212	-9.772	<2e-16 ***
BMI	1.7564	0.1031	17.042	<2e-16 ***
---				
Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

Residual standard error: 5.097 on 247 degrees of freedom  
 Multiple R-squared: 0.5404, Adjusted R-squared: 0.5385  
 F-statistic: 290.4 on 1 and 247 DF, p-value: < 2.2e-16

Der Fit is jedoch with  $R^2 = 0.53$  resp.  $R^2 = 0.54$  in beiden Fällen nur mäßig.

Selbst with allen Datenpunkten and allen Regressoren wird maximal  $R^2 = 0.75$  erreicht:

---

Input

```
lm.fullres <- lm(body.fat ~ age + BMI + neck + chest +
  abdomen + hip + thigh + knee + ankle +
  bicep + forearm + wrist + weightkg + heightcm,
  data = fat)
summary(lm.fullres)
```

```
regsubsets@regsubsets$textit
Call:
lm(formula = body.fat ~ age + BMI + neck + chest + abdomen +
    hip + thigh + knee + ankle + bicep + forearm + wrist + weightkg +
    heightcm, data = fat)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.0761 -2.6118 -0.1055  2.8993  9.2691 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -50.804727  36.489198 -1.392  0.16513  
age          0.061005   0.029862  2.043  0.04217 *  
BMI          0.782993   0.733562  1.067  0.28688  
neck         -0.439082  0.218157 -2.013  0.04528 *  
chest        -0.040915  0.098266 -0.416  0.67751  
abdomen       0.866361  0.085550 10.127 < 2e-16 *** 
hip           -0.206231  0.136298 -1.513  0.13159  
thigh         0.246127  0.135373  1.818  0.07031 .  
knee          -0.005706  0.229564 -0.025  0.98019  
ankle         0.135779  0.208314  0.652  0.51516  
bicep         0.149100  0.159807  0.933  0.35177  
forearm        0.409032  0.186022  2.199  0.02886 *  
wrist          -1.514111  0.493759 -3.066  0.00242 ** 
weightkg      -0.389753  0.221592 -1.759  0.07989 .  
heightcm       0.187196  0.199854  0.937  0.34989 

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.991 on 237 degrees of freedom
Multiple R-squared:  0.7497,    Adjusted R-squared:  0.7349 
F-statistic:  50.7 on 14 and 237 DF,  p-value: < 2.2e-16
```

Dies ist ein Modell mit 15 Koeffizienten. Das Modell ist so komplex, dass es kaum zu interpretieren ist, und man wird versuchen, das Modell zu reduzieren. Anstelle "von Hand" nach einfacheren Modellen zu suchen, kann dieser Prozess automatisiert werden. Dazu dient die Funktion `regsubsets()` in `library(leaps)`. Der quadratische Fehler (resp. das Bestimmtheitsmaß  $R^2$ ) muss dabei modifiziert werden: der quadratische Fehler wird minimiert, wenn wir alle Regressoren ins Modell aufnehmen, also immer im vollen Modell. Zur Modellwahl benutzt man Varianten des quadratischen Fehlers (resp. des Bestimmtheitsmaßes  $R^2$ ), die für die Anzahl der Parameter adjustiert sind.

a:ch04:141

<b>Aufgabe 4.9</b>	
*	<p>Benutzen Sie <code>library(leaps)</code></p> <pre>lm.reg &lt;- regsubsets(body.fat age + BMI + neck + chest + abdomen + hip + thigh + knee + ankle + bicep + forearm + wrist + weightkg + heightcm, data = fat)</pre> <p>and inspizieren Sie das Resultat mit</p> <pre>summary(lm.reg) plot(lm.reg, scale = "r2") plot(lm.reg, scale = "bic") plot(lm.reg, scale = "Cp")</pre> <p>Hinweis: siehe <code>help(plot.regsubsets)</code></p>
*	Benutzen Sie die function <code>leaps()</code> zur Modellselektion.

Allerdings are nun die Werkzeuge, die we in Kapitel 2 kennengelernt haben, unbrauchbar geworden. The statistischen Aussagen in der Zusammenfassung are nur gültig, wenn Modell resp. Hypothesen unabhängig vom Datenmaterial festgelegt are. Wenn anhand des Datenmaterials das Modell erst bestimmt wird, is unklar, wie die geschätzten Koeffizienten verteilt, d.h. wie Konfidenzintervalle zu bestimmen are resp. wie zu testen is. The Software hat keine Information darüber, dass we uns in a Modellwahlprozess befinden and gibt die probabiliyen aus, die bei festem Modell unter Normalverteilungsannahme gelten würden.

Auch die Diagnostik wird unbrauchbar: der zentrale Grenzwertsatz sorgt dafür, dass unter schwachen Unabhängigkeitssannahmen die Residuen bei der großen Anzahl von Termen approximativ normalverteilt are, selbst wenn dies for die Fehler nicht zutrifft.

We are in einer Sackgasse.

We illustrieren nun einen anderen Zugang, der etwas weiter führt. Dazu versetzen we uns an den Anfang der Analyse, nach der ersten Inspektion and Datenkorrektur. Damit we nicht in das oben gesehen Problem laufen, dass die statistischen distributions durch vorhergehende Modellwahlschritte beeinflusst werden, teilen we den Datensatz auf. Einen Teil benutzen we als Trainingsteil, an dem we das Modell wählen and verschiedenen Alternativmöglichkeiten durchspielen können. Der Rest wird als Auswertungsteil reserviert. Dessen Information wird erst nach Modellwahl for die statistische Analyse benutzt.

Bei genauerer Überlegung zeigt sich, dass der Modellwahlschritt nur for die Abschätzung der Fehler kritisch is, nicht for die parameter-Schätzung. Wird der Fehler anhand der Daten geschätzt, die zur Modellwahl benutzt are, so unterschätzen we tendenziell die Fehler. Der Auswertungsteil dient der verlässlichen Fehlerabschätzung and Residuendiagnostik. Dies is eine eingeschränkte Aufgabe. Deshalb reservieren we dafür nur einen kleineren Teil.

---

```
sel <- runif(dim(fat)[1])
fat$train <- sel < 2/3
rm(sel)
```

*Input*

The Ausreisser eliminieren we aus dem Trainingsteil

---

```
fat$train[c(39, 41, 216)] <- FALSE
summary(fat$train)
```

*Input*

			Output
Mode	FALSE	TRUE	
logical	89	163	

Unsere Zielvariable is `body.fat`, or, proportional dazu,  $1/density$ .

We versuchen zunächst, die Variablen inhaltlich zu sortieren. for die Dichte habe we eine physikalische Definition

$$Dichte = \frac{Gewicht}{Volumen}.$$

Unter den Variablen, die als Regressoren in Betracht kommen, haben we eine Variable, die direkt das Gewicht angibt (`weight` resp. `weightkg`), eine ganze Reihe von Variablen, die Körpermaße widerspiegeln, sowie das Alter `age`.

Aus der gemessenen Dichte und dem gemessenen Gewicht lässt sich das Volumen errechnen. We erweitern dadurch die Variablen. Da we hier nur eine Gewichtsmessung pro Person haben, bleibt kein Platz for personenbezogene Statistik.

---

`fat$vol <- fat$weightkg/fat$density`

---

We versuchen nun, das Volumen `fat$vol` zu schätzen. The Körpermaße are lineare values. In einer groben Approximation können we daraus Volumen-values ableiten. The einzige Längen-information, die we haben, steckt in `height`. Mangels besserer Information nehmen we an, dass alle Körperteile eine Länge haben, die proportional zur Körpergröße is. We zielen auf ein lineares Modell. Deshalb können we lineare Faktoren vernachlässigen. Approximieren we die Körperteile durch Zylinder, so erhalten we, bis auf lineare Faktoren

---

`fat$neckvol <- fat$neck^2 * fat$heightcm`

---

`fat$chestvol <- fat$chest^2 * fat$heightcm`

---

`fat$abdomenvol <- fat$abdomen^2 * fat$heightcm`

---

`fat$hipvol <- fat$hip^2 * fat$heightcm`

---

`fat$thighvol <- fat$thigh^2 * fat$heightcm`

---

`fat$kneevol <- fat$knee^2 * fat$heightcm`

---

`fat$anklevol <- fat$ankle^2 * fat$heightcm`

---

`fat$bicepvol <- fat$bicep^2 * fat$heightcm`

---

`fat$forearmvol <- fat$forearm^2 * fat$heightcm`

---

`fat$wristvol <- fat$wrist^2 * fat$heightcm`

---

Als nächstes untersuchen we die interne Struktur der Regressor-Kandidaten im Trainingsteil. We tun dies getrennt for die linearen Variablen and for die Volumen- Variablen. Dazu benutzen we die function `prcomp()`, die zu gegebenen Variablen schrittweise beste lineare Prediktoren liefert.

`fatprincomp`

for die approximativen Körperteil-Volumen are die Hauptkomponenten:

---

`pcfatvol <- prcomp(fat[, 20:29], subset = fat$train)`

---

`round(pcfatvol$rotation, 3)`

---

	Output							
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
neckvol	0.054	-0.017	0.063	-0.070	0.570	0.045	0.787	-0.169
chestvol	0.548	0.457	0.691	0.012	-0.113	-0.004	-0.003	-0.012
abdomenvol	0.650	0.290	-0.697	-0.059	0.049	-0.011	-0.017	0.016
hipvol	0.483	-0.748	0.099	0.437	-0.031	0.069	0.003	0.018
thighvol	0.185	-0.370	0.074	-0.870	-0.244	-0.017	0.079	0.012
kneevol	0.056	-0.082	0.055	-0.061	0.336	-0.822	-0.278	-0.159
anklevol	0.016	-0.032	0.034	-0.005	0.140	-0.236	-0.047	-0.120
bicepvol	0.051	-0.047	0.079	-0.180	0.552	0.508	-0.531	-0.333
forearmvol	0.024	-0.019	0.074	-0.086	0.388	0.019	-0.104	0.906
wristvol	0.009	-0.005	0.017	0.002	0.110	-0.053	0.044	0.003
	PC9	PC10						
neckvol	0.067	0.093						
chestvol	0.005	0.002						
abdomenvol	-0.015	-0.002						
hipvol	0.011	0.004						
thighvol	-0.015	-0.019						
kneevol	0.299	0.056						
anklevol	-0.949	0.073						
bicepvol	0.030	0.010						
forearmvol	-0.049	0.044						
wristvol	-0.048	-0.990						

Das Muster der Vorzeichen bei den Ladungen gibt Hinweise auf die interne Struktur. The erste Hauptkomponente  $PC1$  is eine Linearkombination von Variablen, die im wesentlichen den Torso beschreiben. The zweite Hauptkomponente kontrastiert den Oberkörper bis zum Bauch with den unteren Teil des Torsos. The dritte unterscheidet das Bauchvolumen vom Rest des Torsos.

a:ch04:15

**Aufgabe 4.10** Skizzieren Sie für die nachfolgenden Komponenten  $PC4, \dots, PC10$ , welche Körpergeometrie durch sie beschrieben wird.

Der Versuch, das errechnete Volumen durch die approximativen Körperteil-Volumen darzustellen, resultiert in für den Trainingsteil ein hohes Bestimmtheitsmaß.

```
lm.vol <- lm(vol ~ neckvol + chestvol + abdomenvol +
               hipvol + thighvol + kneevol +
               anklevol + bicepvol + forearmvol +
               wristvol,
               data = fat, subset = fat$train)
summary(lm.vol)
```

Call:  
lm(formula = vol ~ neckvol + chestvol + abdomenvol + hipvol +  
thighvol + kneevol + anklevol + bicepvol + forearmvol + wristvol,

```

data = fat, subset = fat$train)

Residuals:
    Min     1Q Median     3Q    Max 
-9.3561 -1.0383  0.1005  1.1208  4.3312 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2.264e+00  1.454e+00  1.557 0.121648  
neckvol     1.800e-05  9.975e-06  1.805 0.073132 .  
chestvol    9.390e-06  1.563e-06  6.008 1.34e-08 *** 
abdomenvol  1.177e-05  1.284e-06  9.161 3.31e-16 *** 
hipvol      8.418e-06  2.071e-06  4.065 7.67e-05 *** 
thighvol    1.390e-05  3.726e-06  3.731 0.000269 *** 
kneevol     -1.954e-06 9.813e-06 -0.199 0.842457  
anklevol    2.749e-05  1.115e-05  2.465 0.014809 *  
bicepvol   3.329e-05  9.224e-06  3.609 0.000416 *** 
forearmvol  2.362e-05  1.195e-05  1.977 0.049869 *  
wristsvol  -2.002e-05 4.483e-05 -0.446 0.655904  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.959 on 152 degrees of freedom
Multiple R-squared:  0.9748,    Adjusted R-squared:  0.9732 
F-statistic: 588.3 on 10 and 152 DF,  p-value: < 2.2e-16

```

Bei Einbeziehung auch der linearen Variablen können we im Trainingsteil des Bestimmtheitsmaß nur geringfügig erhöhen.

Mit den Functions aus `library(leaps)` kann wieder automatisch nach “optimalen” Modellen gesucht werden.

---

*Input*

```

library(leaps)
l1 <- leaps(x = fat[, c(6:15, 20:29)], y = fat$vol)

```

Wenn we versuchen wollen, nicht das Volumen zu schätzen, sondern den Fettanteil als Linearkombination der entsprechenden Komponenten darzustellen, können we die entsprechenden Hilfsvariablen konstruieren.

---

*Input*

```

fat$neckvolf <- fat$neckvol / fat$weightkg
fat$chestvolf <- fat$chestvol / fat$weightkg
fat$abdomenvolf <- fat$abdomenvol / fat$weightkg
fat$hipvolf <- fat$hipvol / fat$weightkg
fat$thighvolf <- fat$thighvol / fat$weightkg
fat$kneevolf <- fat$kneevol / fat$weightkg
fat$anklevolf <- fat$anklevol / fat$weightkg
fat$bicepvolf <- fat$bicepvol / fat$weightkg
fat$forearmvolf <- fat$forearmvol / fat$weightkg
fat$wristsvolf <- fat$wristsvol / fat$weightkg

```

We begin with a einfachen Modell. We benutzen nur eine Variable (`abdomenvolf`) aus der Gruppe der Variablen, die den Torso beschreibt, and eine der Variablen (`wristvolf`) aus den höheren Hauptkomponenten. Damit erreichen we fast die Genauigkeit des ersten Modells with dem vollen Variablenatz.

```
lm.volf <- lm(body.fat ~ abdomenvolf + wristvolf, data = fat, subset = fat$train) Input
summary(lm.volf)

Output
Call:
lm(formula = body.fat ~ abdomenvolf + wristvolf, data = fat,
subset = fat$train)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.3306 -3.2095  0.3383  2.9397  9.1401 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.5439910  6.5135540   0.237   0.813    
abdomenvolf 0.0022769  0.0001907  11.940 < 2e-16 ***  
wristvolf   -0.0345820  0.0054366  -6.361 2.01e-09 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.266 on 160 degrees of freedom
Multiple R-squared:  0.6608,    Adjusted R-squared:  0.6565 
F-statistic: 155.8 on 2 and 160 DF,  p-value: < 2.2e-16
```

a:ch04:16

<b>Aufgabe 4.11</b>	
*	Ergänzen Sie die Variablen durch andere volumenbezogene Variable in dem obigen Modell. gewinnen Sie an Schätzgenauigkeit?
**	Versuchen Sie, die Variable <code>age</code> in die Modellierung with einzubeziehen. Wie berücksichtigen Sie <code>age</code> im Modell?
**	The function <code>mvr()</code> in <code>library(pls)</code> steht bereit, um Regressionen auf der Basis der Hauptkomponenten durchzuführen. Benutzen Sie die funktion zur Regression. Wie unterscheidet sie sich von der gewöhnlichen Kleinstquadrat-Regression?

Zur Konstruktion des Modells haben we den Trainingsteil benutzt. The Genauigkeit des so gewonnenen Modells überprüfen we nun am Auswertungsteil. Dazu benutzen we die function `predict.lm()`, die ein with `lm()` geschätztes lineares Modell auf einen neuen Datensatz anwendet. Z.B.

*Input*

`predict.lm@predict.lm|tex  
lm@lm|textit`

```
image@image|textit
fat.eval <- fat[fat$train == FALSE, ]
pred <- predict.lm(lm.volf, fat.eval, se.fit = TRUE)
```

a:ch04:17

<b>Aufgabe 4.12</b>	
*	Schätzen Sie die Genauigkeit des Modells durch die Daten des Auswertungsteils.
*	Führen Sie die eine Diagnostik des gewonnen Modells anhand der Daten des Auswertungsteils durch.

## 4.8 High Dimensions

Probleme in kleinen Dimensionen können wir umfassend darstellen und analysieren. Höhere Dimensionen erfordern es oft, eine spezielle Analyse-Strategie zu entwerfen. The formale Anwendung von Standard-Methoden kommt hier schnell an ihre Grenzen.

Höhere Dimensionen, etwa von 10 bis 100, are in vielen Anwendungsbereichen üblich. Aber auch Probleme in großen Dimensionen are alltäglich. Wir müssen uns darüber im Klaren sein, dass die Dimension eine Frage der Modellierung is, nicht nur eine Frage des Problems. Digitales Video (DV PAL) zum Beispiel zeichnet Bilder im Format  $720 \times 576$  auf. Ein einzelnes Bild with drei Farben gibt also einen vector im  $720 \times 576 \times 3 = 1244160$ -dimensionalen Raum, jede Sekunde Video das 25-fache. Haben wir Bilddaten zu bearbeiten, so is es unsere Wahl, ob we die Bildbearbeitung als Problem with Dimension  $d = 1244160$ , betrachten, or als sequence von 1244160 (nicht unabhängigen!) observations with  $d = 1$ .

Beim Übergang von  $d = 1244160$  zu  $d = 1$  verlagern we Information, die implizit in den Dimensionen steckt, in Strukturinformation, die folglich modelliert werden muss.

Als Anmerkung: in der Praxis geht man einen Mittelweg. Man zerlegt das Bild in Blöcke, z.B. der Größe  $64 \times 64$ . Pixel innerhalb eines Blockes werden simultan behandelt. The Blöcke werden sequentiell behandelt - sichtbar bei der nächsten Störung im Fernsehen.

Bei hochdimensionalen Problem is die Statistik oft gar nicht sichtbar - sie is versteckt in der Hardware als "imbedded system".

**fig:cDna**  
The Abbildung 4.6 is ein Beispiel aus einer Analyse with R for einen hochdimensionalen Datensatz vom Mikroarray-cDNA-Daten (news:Sawitzki:2002). Ein einzelnes Datum in diesem Datensatz besteht aus Messungen an 4227 Proben with jeweils 4 Teilmessungen (*fg.green*, *fg.red*, *bg.green*, *bg.red*). The wesentliche function, die hier zur Visualisierung benutzt wird, is *image()*, with der eine Variable *z* anhand einer Farbtabelle gegen zwei Koordinaten *x*, *y* dargestellt werden kann. Dargestellt is eine Beobachtung. The vier Kanäle der Teilmessungen are nebeneinander gestellt.

The Farben codieren das Ergebnis einer Voranalyse - die roten Punkte signalisieren Problemzonen auf dem cDNA-Chip. In diesem Fall kann aus dem Muster der Selektion ein spezifisches Problem in der Fertigung identifiziert werden.

Themenorientierte Übersichten über R-Pakete, insbesondere auch zu multivariaten Problemen, are in <<http://cran.at.r-project.org/src/contrib/Views/>> zu finden.

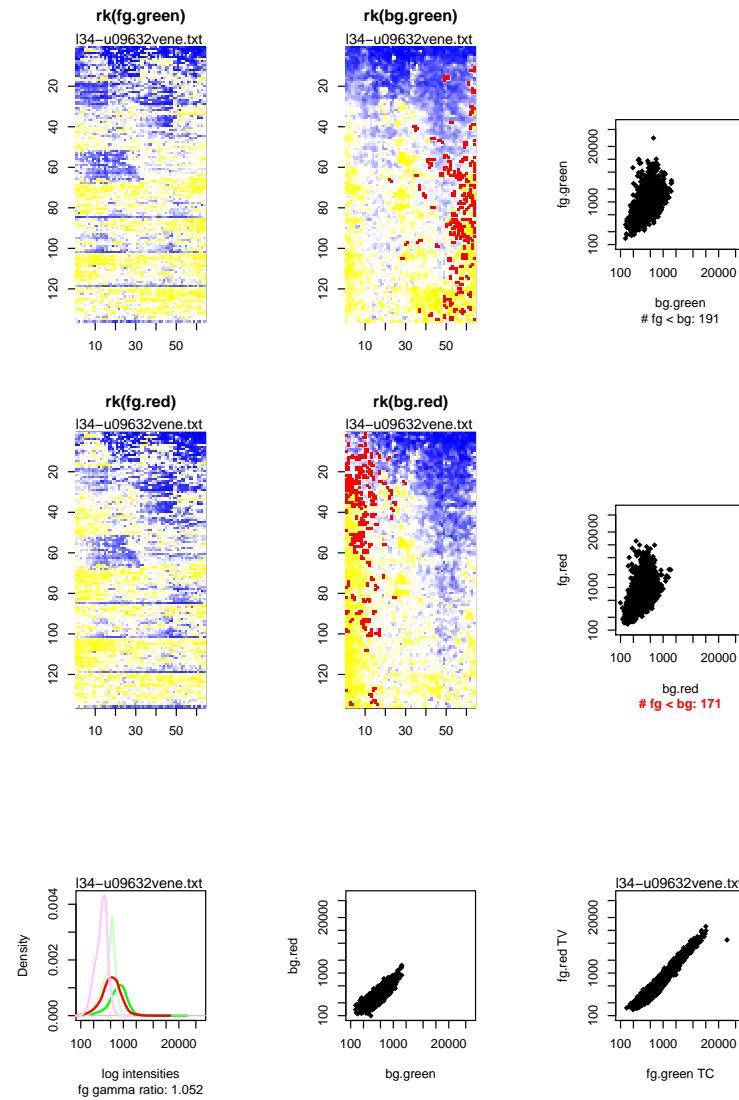
Figure 4.6 Ein  $4227 \times 4$  Datum aus a Microarray-Experiment

fig:cDna

## 4.9 Statistical Summary

The Analyse multivariater Daten konnte in diesem Zusammenhang nur gestreift werden.<sup>Multivariate Probleme tauchen implizit schon bei Regressionsproblemen auf (siehe Kapitel 2). Bei den einfachen Regressionsproblemen bezogen sich die multivariaten Aspekte aber nur auf deterministische parameter. Im allgemeinen Fall haben wir aber eine multivariate statistische distribution zu analysieren. An dieser Stelle muss die Einführung abbrechen, und weiteres bleibt weiterführenden Vorlesungen vorbehalten.</sup>

## 4.10 Literature and Additional References

Generated by Sweave from:  
+ Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S04multiv.Rnw.tex,v +  
+ Revision: 1.1 +  
+ Date: 2008/02/14 18:48:39 +  
+ name: +  
+ Author: j40 +  
+Source : /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S04multiv.Rnw.tex, v +  
+Revision : 1.1 +  
+Date : 2008/02/14 18:48:39 +  
+name : +  
+Author : j40 +

# R as a Programming Language and Environment

ch:0A

R is eine interpretierte Ausdruckssprache. Ausdrücke are zusammengesetzt aus Objekten and operators.

## A.11 Help and Information

R <i>help</i>	
<i>help()</i>	Information über ein Objekt/eine function <i>Beispiel:</i> <code>help(help)</code>
<i>args()</i>	Zeigt Argumente einer function
<i>example()</i>	Führt evtl. vorhandene examples aus <i>Beispiel:</i> <code>example(plot)</code>
<i>help.search()</i>	Sucht Information über ein Objekt/eine function
<i>apropos()</i>	Lokalisiert nach Stichwort
<i>demos()</i>	Führt Demos zu a Themenbereich aus <i>Beispiel:</i> <code>demos(graphics)</code> <i>demos()</i> listet die zur Verfügung stehenden Themenbereiche

## A.12 Names and Search Paths

objects werden durch names identifiziert. Anhand des name werden objects in einer Kette von Suchbereichen identifiziert. The aktuellen Suchbereiche können with `search()` inspiziert werden.

R <i>Suchpfade</i>	
	(Fortsetzung)→

<code>mode@mode   text length@length   typeof@typeof  </code>	<b>R Suchpfade</b> (Fortsetzung)	
	<b>search()</b>	Liste der aktuellen Suchbereiche, beginnend with <code>.GlobalEnv</code> bis hinab zum Basis-Paket <code>package:base</code> . <i>Beispiel:</i> <code>search()</code>
	<b>searchpaths()</b>	Liste der Zugriffspfade zu aktuellen Suchbereichen <i>Beispiel:</i> <code>searchpaths()</code>
	<b>objects()</b>	Liste der objects in a Suchbereich <i>Beispiele:</i> <code>objects()</code> <code>objects("package:base")</code>
	<b>ls()</b>	Liste der objects in a Suchbereich <i>Beispiele:</i> <code>ls()</code> <code>ls("package:base")</code>
	<b>ls.str()</b>	Liste der objects und ihrer Struktur in a Suchbereich <i>Beispiele:</i> <code>ls.str()</code> <code>lsf.str("package:base")</code>
	<b>find()</b>	Lokalisiert nach Stichwort. Findet auch überlagerte Einträge <i>Aufruf:</i> <code>find(what, mode = "any", numeric = FALSE, simple.words = TRUE)</code>
	<b>apropos()</b>	Lokalisiert nach Stichwort. Findet auch überlagerte Einträge <i>Aufruf:</i> <code>apropos(what, where = FALSE, ignore.case = TRUE, mode = "any")</code>

Functions können sowohl bei Definition als auch bei Aufruf geschachtelt sein. Dies macht eine Erweiterung der Suchpfade nötig. The dynamische Identifikation von Objekten benutzt Umgebungen (environments), um in Functions lokale Variable or globale Variablen aufzulösen.

<b>R Suchpfade (Fortsetzung)</b>	
<b>environment()</b>	Aktuelle Auswertungsumgebung <i>Beispiel:</i> <code>environment()</code>
<b>sys.parent()</b>	Vorausgehende Auswertungsumgebungen <i>Beispiel:</i> <code>sys.parent(1)</code>

objects haben zwei implizite Attribute, die erfragt werden with `mode()` and `length()`. The function `typeof()` gibt den (internen) Speichermodus eines Objektes an.

Ein `class`-Attribut benennt die Klasse eines Objektes.

### A.13 Customizing

R bietet eine Reihe von Möglichkeiten, das System zu konfigurieren, so dass beim Start and beim Ende bestimmte Kommandos ausgeführt werden. Falls vorhanden, werden beim Start die Dateien `.Rprofile` and `.RData` eingelesen and ausgevaluest. Details können system-spezifisch sein. The jeweils spezifische Information erhält man with `help(Startup)`.



### A.14 Basic Data Types

s:A.1

R Basis-Daten-typen	
<i>numeric</i>	<b>real</b> or <b>integer</b> . In R: real is stets doppelt-genau. Einfache Genauigkeit wird für externe Aufrufe zu anderen Sprachen wie .C or .FORTRAN unterstützt. Functions wie <code>mode()</code> and <code>typedef()</code> können je nach Implementierung auch den Speicherungsmodus (single, double ...) melden. <i>Beispiele:</i> 1.0 2 3.14E0
<i>complex</i>	komplex, in cartesischen Koordinaten <i>Beispiel:</i> 1.0+0i
<i>logical</i>	TRUE, FALSE. In R: auch vordefinierte Variable T, F. In S-Plus are T and F Basis-objects.
<i>character</i>	Zeichenketten. Delimiter are alternativ " or '. <i>Beispiel:</i> "T", 'klm'
<i>list</i>	Allgemeine Liste. The Listenelemente können auch von unterschiedlichem Typ sein. <i>Beispiel:</i> list(1:10, "Hello")
<i>function</i>	R-function <i>Beispiel:</i> sin
<i>NULL</i>	Spezialfall: leeres Objekt <i>Beispiel:</i> NULL

Zusätzlich zu den Konstanten TRUE and FALSE gibt es drei spezielle values for Ausnahmesituationen:

spezielle Konstan-ten	
<b>TRUE</b>	Alternativ: <i>T</i> . Typ: logical.
<b>FALSE</b>	Alternativ: <i>F</i> . Typ: logical.
<b>NA</b>	"not available". Typ: logical. NA is von TRUE and FALSE verschieden

(Fortsetzung)→

<i>spezielle Konstanten</i> (Fortsetzung)	
<i>NaN</i>	"not a valid numeric value". Implementationsabhängig. Sollte dem IEEE Standard 754 entsprechen. Typ: numeric. <i>Beispiel:</i> 0/0
<i>Inf</i>	unendlich. Implementationsabhängig. Sollte dem IEEE Standard 754 entsprechen. Typ: numeric. <i>Beispiel:</i> 1/0

**A.15 Output for Objects****polymorph**

The Objekt-Attribute and weitere Eigenschaften können abgefragt or with Ausgaberoutinen angefordert werden. The Ausgaberoutinen are in der Regel ***polymorph***, d.h. sie erscheinen in Varianten, die den jeweiligen Objekten angepasst werden.

R <i>Inspektion</i>	
<i>print()</i>	Standard-Ausgabe
<i>structure()</i>	Ausgabe, optional with Attributen
<i>summary()</i>	Standard-Ausgabe als Übersicht, insbesondere for Modellanpassungen
<i>plot()</i>	Standard-Grafikausgabe



### A.16 Object Inspection

The folgende Tabelle fasst die wichtigsten Informationsmöglichkeiten über objects zusammen.

<i>Inspektion von Objekten</i>	
<b><code>str()</code></b>	Stellt die interne Struktur eines Objekts in kompakter Form dar. <i>Aufruf:</i> <code>str(&lt;object&gt;)</code>
<b><code>structure()</code></b>	Stellt die interne Struktur eines Objekts dar. Dabei können Attribute für die Darstellung als Parameter übergeben werden. <i>Beispiel:</i> <code>structure(1:6, dim = 2:3)</code> <i>Aufruf:</i> <code>structure(&lt;object&gt;, ...)</code>
<b><code>class()</code></b>	Objekt-Klasse. Bei neueren Objekten ist die Klasse als Attribut gespeichert. In älteren S or R-Versionen ist sie durch Typ und andere Attribute implizit bestimmt.
<b><code>mode()</code></b>	Modus (Typ) eines Objekts.
<b><code>storage.mode()</code></b>	Speichermodus eines Objekts.
<b><code>typeof()</code></b>	Modus eines Objekts. Kann vom Speichermodus abweichen. Je nach Implementierung kann etwa eine numerische Variable standardmäßig doppelt- oder einfach genau abgespeichert werden.
<b><code>length()</code></b>	Länge = Anzahl der Elemente
<b><code>attributes()</code></b>	Liest/setzt Attribute eines Objekts, wie z.B. names, Dimensionen, Klassen.
<b><code>names()</code></b>	names-Attribut für Elemente eines Objekts, z.B. eines Vektors. <i>Aufruf:</i> <code>names(&lt;obj&gt;)</code> gibt das names-Attribut von <code>&lt;obj&gt;</code> . <code>names(&lt;obj&gt;)&lt;-&lt;charvec&gt;</code> setzt es. <i>Beispiel:</i> <code>x&lt;-values</code> <code>names(x)&lt;- &lt;charvec&gt;</code>



**A.17 System Inspection**

```
trellis.par.set@trellis.par.set
lattice.options@lattice.options
```

The folgende Tabelle fasst die wichtigsten Informationsmöglichkeiten über die allgemeine Systemumgebung zusammen.

<i>System-Inspektion</i>	
<code>search()</code>	aktueller Suchpfad
<code>ls()</code>	aktuelle objects
<code>methods()</code>	generische Methoden <i>Aufruf:</i> <code>methods(&lt;fun&gt;)</code> zeigt spezialisierte Funktionen zu <code>&lt;fun&gt;</code> , <code>methods(class = &lt;c&gt;)</code> die klassenspezifischen Functions zu class <code>&lt;c&gt;</code> . <i>Beispiele:</i> <code>methods(plot)</code> <code>methods(class = lm)</code>
<code>data()</code>	zugreifbare Daten
<code>library()</code>	zugreifbare Bibliotheken
<code>help()</code>	allgemeines help-System
<code>options()</code>	globale Optionen
<code>par()</code>	parameter-Einstellungen des Grafik-Systems

The Optionen des `lattice`-Systems können with `trellis.par.set()` resp. `lattice.options()` kontrolliert werden.

R is im umgebenden Betriebssystem verankert. Einige Variable, wie z.B. Zugriffspfade, Zeichen-codierung etc. werden von dort übernommen.

<i>System-Umgebung</i>	
<code>getwd()</code>	aktuelleres Arbeitsverzeichnis
<code>setwd()</code>	setzt aktuelles Arbeitsverzeichnis
<code>dir()</code>	listet Dateien im aktuellen Arbeitsverzeichnis
<code>system()</code>	ruft System-Functions auf



### A.18 Complex Data Types

The Interpretation von Basistypen or abgeleiteten Typen kann durch ein or mehrere `class`-Attribute spezifiziert werden. Polymorphe Functions wie `print` or `plot` valuesn dieses Attribut aus and rufen nach Möglichkeit entsprechend der Klasse spezialisierte Varianten auf (Siehe [ch02:polymorphe:polymorph](#) 2.6.5Seite 2-42).

Zur Speicherung von Datumsangaben and Zeiten stehen entsprechende Klassen bereit. Nähere Information zu diesen Datentypen erhält man with

`help(DateTimeClasses)`.

R is vektor-basiert. Einzelne Konstanten or values are nur vectors der speziellen Länge 1. Sie genießen keine Sonderbehandlung.

Zusammengesetzte Objekttypen	
<b>vectors</b>	R Basis-Datentypen
<b>Matrizen</b>	vectors with zwei-dimensionalem Layout
<b>Arrays</b>	<p>vectors with höherdimensionalem Layout</p> <p><code>dim()</code> definiert Dimensionsvektor</p> <p><code>Beispiel:</code> <code>x &lt;- runif(100)</code>  <code>dim(x) &lt;- c(5, 5, 4)</code></p> <p><code>array()</code> konstruiert neuen vector with gegebener Dimensionsstruktur</p> <p><code>Beispiel:</code> <code>z &lt;- array(0, c(4, 3, 2))</code></p> <p><code>rbind()</code> kettet Reihen an</p> <p><code>cbind()</code> kettet Spalten an</p>
<b>Faktoren</b>	<p>Sonderfall for kategorioelle Daten</p> <p><code>factor()</code> wandelt vector in Faktor um</p> <p><code>Siehe auch</code> <a href="#">Abschnitt 2.2.1</a></p> <p><code>ordered()</code> wandelt vector im Faktor with geordneten Stufen um. Dies is eine Abkürzung for <code>factor(x, ..., ordered = TRUE)</code></p> <p><code>levels()</code> gibt die Stufen eines Faktors an</p> <p><code>Beispiel:</code> <code>x &lt;- c("a", "b", "a", "c", "a")</code>  <code>xf &lt;- factor(x)</code>  <code>levels(xf)</code>      resutls in  <code>[ 1 ] "abc"</code></p> <p><code>tapply()</code> wendet eine function getrennt for alle Stufen von Faktoren einer Faktorliste an</p>

(Fortsetzung)→

<b>Zusammengesetzte Objekttypen</b> (Fortsetzung)	
<b>Listen</b>	Analog vectors, with Elementen auch unterschiedlichen Typs  <code>list()</code> erzeugt Liste  <code>Aufruf:</code> <code>list(&lt;Komponenten&gt;)</code>  <code>[[ ]]</code> Indexweiser Zugriff auf Komponenten  <code>Liste\$Komponente</code> Zugriff nach names  <code>Beispiel:</code> <code>l &lt;- list(name = "xyz", age = 22, fak = "math")</code> <code>&gt;l[[2]]</code> <code>22</code> <code>&gt;l\$age</code> <code>22</code>
<b>Datenrahmen</b>	<b>data frames</b> Analog Arrays resp. Listen, with spaltenweise einheitlichem Typ und einheitlicher Spaltenlänge  <code>data.frame()</code> analog <code>list()</code> , aber Restriktionen müssen erfüllt sein  <code>attach()</code> fügt Datenrahmen in die aktuelle Suchliste ein, d.h. for Komponenten reicht der Komponentenname.  <code>detach()</code>

**FixMe:**

Daten-  
rahmen  
vs da-  
ta.frames

### A.19 Accessing Components

A.2

The Länge von vectors is ein dynamisches Attribut. Sie wird bei Bedarf erweitert and gekürzt. Insbesondere gilt implizit eine “Recycling-Regel”: Hat ein vector nicht die erforderliche Länge for eine Operation, so wird er periodisch bis zur erforderlichen Länge wiederholt.

Auf vector-Komponenten kann über Indizes zugegriffen werden. The Indizes können explizit or als Regel-Ausdruck angegeben werden.

R Index-Zugriff	
<code>x[&lt;indices&gt;]</code>	Indizierte Komponenten von <i>x</i> <i>Beispiel:</i> <code>x[1:3]</code>
<code>x[-&lt;indices&gt;]</code>	<i>x</i> ohne indizierte Komponenten <i>Beispiel:</i> <code>x[-3]</code> <i>x</i> ohne 3. Komponente
<code>x[&lt;condition&gt;]</code>	Komponenten von <i>x</i> , for die <code>&lt;condition&gt;</code> gilt. <i>Beispiel:</i> <code>x[x &lt; 0.5]</code>

vectors (and andere objects) können auf höherdimensionale Konstrukte abgebildet werden. The Abbildung wird durch zusätzliche Dimensions-Attribute beschrieben. Nach Konvention erfolgt eine spaltenweise Einbettung, d.h. der erste Index variiert zuerst (FORTRAN-Konvention). operators and Functions können die Dimensions-Attribute ausvaluesn.

R Index-Zugriff	
<code>dim()</code>	Setzt or liest die Dimensionen eines Objekts <i>Beispiel:</i> <code>x &lt;- 1:12 ; dim(x) &lt;- c(3, 4)</code>
<code>dimnames()</code>	Setzt or liest names for die Dimensionen eines Objekts
<code>nrow()</code>	Gibt die Anzahl der Zeilen = Dimension 1
<code>ncol()</code>	Gibt die Anzahl der Spalten = Dimension 2
<code>matrix()</code>	Erzeugt eine Matrix with vorgegebenen Spezifikationen <i>Aufruf:</i> <code>matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)</code> Siehe auch example <a href="#">expl:ch01-MC06 ch01-MC06</a> II.8 (page II-23)
<code>array()</code>	Erzeugt eine evtl. höherdimensionale Matrix <i>Beispiel:</i> <code>array(x, dim = length(x), dimnames = NULL)</code>

R Array-Zugriffe	
<code>cbind()</code> <code>rbind()</code>	Verkettet Zeilen resp. Spalten
<code>split()</code>	Teilt einen vector nach Faktoren auf
<code>table()</code>	Erzeugt eine Tabelle von Besetzungsnumbers

R Iteratoren	
<code>apply()</code>	wendet eine function auf die Zeilen or Spalten einer Matrix an Aufruf: <code>apply(x, MARGIN, FUNCTION, ...)</code> Margin = 1: Zeilen, Margin = 2: Spalten. Siehe auch <a href="#">example 1.8 (page II-23)</a>
<code>lapply()</code>	wendet eine function auf die Elemente einer Liste an Aufruf: <code>lapply(X, FUN, ...)</code>
<code>sapply()</code>	wendet eine function auf die Elemente einer Liste, eines Vektors or einer Matrix an. Falls mögliche werden Dimensionsnames übernommen. Aufruf: <code>sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)</code>
<code>tapply()</code>	wendet eine function auf Komponenten eines Objekts in Abhängigkeit von einer Liste von kontrollierenden Faktoren an.
<code>by()</code>	Objekt-orientierte Variante von <code>tapply</code> Aufruf: <code>by(data, INDICES, FUN, ...)</code>
<code>aggregate()</code>	Berechnet Statistiken for Teilmengen Aufruf: <code>aggregate(x, ...)</code>
<code>replicate()</code>	valuest eine Ausdruck wiederholt aus (z. Bsp. with Erzeugung von random numbers zur Simulation). Aufruf: <code>replicate(n, expr, simplify = TRUE)</code>
<code>outer()</code>	erzeugt eine Matrix with allen Paar-Kombinationen aus zwei vectors, and wendet eine function auf jedes Paar an. Aufruf: <code>outer(vec1, vec2, FUNCTION, ...)</code>

### A.20 Data Manipulation

<i>Transformationen</i>	
<code>seq()</code>	Erzeugt eine Sequenz
<code>abbreviate()</code>	

<i>Transformationen</i>	
<code>duplicated()</code>	Prüft auf mehrfach auftretende values
<code>unique()</code>	Erzeugt vector ohne mehfach auftretende values
<code>match()</code>	Gibt Position eines Werts in a vector
<code>pmatch()</code>	Partielles Matching

<i>Zeichenketten-Transformationen</i>	
<code>casefold()</code>	Wandelt in Klein- or Großbuchstben um
<code>tolower()</code>	Wandelt in Kleinletters um
<code>toupper()</code>	Wandelt in Großletters um
<code>chartr()</code>	Übersetzt Zeichen in a Zeichen-vector
<code>substring()</code>	

<i>Transformationen</i>	
<code>table()</code>	Erzeugt eine Kreuztabelle
<code>expand.grid()</code>	Erzeugt einen Datenrahmen with allen Kombinationen gegebener Faktoren
<code>reshape()</code>	Wandelt zwischen einer Kreuztabelle (Spalte pro Variable) and einer langen Tabelle (Variablen in Zeilen, with zusätzlicher Indikator-Spalte) um
<code>merge()</code>	Kombiniert Datenrahmen

**ToDo:**  
 extend res-  
 tape data  
 frames  
 Harrell  
 4.2.7  
**ToDo:**  
 add mer-  
 ge data  
 frames  
 Harrell  
 4.2.5ff

<i>Transformationen</i>	
<code>t()</code>	Transponiert Zeilen und Spalten <i>Aufruf:</i> <code>t(x)</code>
<code>aperm()</code>	Generalisierte Permutation <i>Aufruf:</i> <code>aperm(x, perm)</code> Dabei ist <code>perm</code> eine Permutation der Indizes von <code>x</code> .
<code>split()</code>	Teilt einen vector nach a Faktor auf
<code>unsplit()</code>	Kombiniert Komponenten zu a vector

**ToDo:**

move to  
cbind etc

## A.21 Operators

s:A.4

Ausdrücke in R können aus Objekten and operators zusammengesetzt sein. The folgende Tabelle is nach Vorrang geordnet (höchster Rang oben).

R Basisoperatoren	
\$	Komponenten-Selektion <i>Beispiel:</i> <code>list\$item</code>
[ [	Indizierung, Elementzugriff <i>Beispiel:</i> <code>x[i]</code>
^	Potenzierung <i>Beispiel:</i> <code>x^3</code>
-	unitäres Minus
:	sequence-Generierung <i>Beispiele:</i> <code>1:5</code> <code>5:1</code>
%<name>%	spezielle operators. Können auch benutzer-definiert sein. <i>Beispiele:</i> <code>"%deg2%"&lt;-function(a, b) a + b^2</code> <code>2 %deg2% 4</code>
* /	Multiplikation, Division
+ -	Addition, Subtraktion
< > < = > = == !=	Vergleichsoperatoren
!	Negation
&   &&	and, or &&,    are “Shortcut”-operators
<- ->	Zuweisung

Haben die Operanden nicht die gleiche Länge, so wird der kürzere Operand zyklisch wiederholt.

operators der Form %<name>% können vom Benutzer definiert werden. The Definition folgt den Regeln for Functions.

Ausdrücke können als sequence with trennendem Semikolon geschrieben werden. Ausdrucksgruppen können durch {...} zusammengefasst werden.



## A.22 Functions

s:A.5

Functions are spezielle objects. Functions können Resultat-objects übergeben.

function|(  
**ToDo:**  
superas-  
signment  
introR  
0.5

R <i>Funktions-deklarationen</i>	
<b>Deklaration</b>	<code>function ( &lt;formale Parameterliste&gt; ) &lt;Ausdruck&gt;</code> <i>Beispiel:</i> <code>fak &lt;- function(n) prod(1:n)</code>
<b>Formale parameter</b>	<code>&lt;Parametername&gt;</code> <code>&lt;Parametername&gt; = &lt;Default-value&gt;</code>
<b>Formale Parameterliste</b>	Liste von formalen Parametern, durch Komma getrennt <i>Beispiele:</i> <code>n, mean = 0, sd = 1</code>
...	Variable Parameterliste. Variable Parameterlisten können innerhalb von Prozeduren weitergegeben werden. <i>Beispiel:</i> <code>mean.of.all &lt;- function (...)mean(c(...))</code>
<b>Funktions-Resultate</b>	<code>return &lt;value&gt;</code> bricht Funktionsauswertung ab und übergibt value
	<code>&lt;value&gt;</code> als letzter Ausdruck in einer Funktionsdeklaration: übergibt value
<b>Funktions-Resultate</b>	<code>&lt;Variable&gt;&lt;-&lt;value&gt;</code> übergibt value. Normalerweise wirken assignments nur auf lokale Kopien der Variablen. The Zuweisung with <code>&lt;-</code> jedoch sucht die Zielvariable in der gesamten Umgebungshierarchie.

R <i>Funktionsaufruf</i>	
<b>Funktionsaufruf</b>	<code>&lt;name&gt;(&lt;Aktuelle Parameterliste&gt;)</code> <i>Beispiel:</i> <code>fak(3)</code>
<b>Aktuelle Parameterliste</b>	values werden zunächst der Position nach zugeordnet. Abweichend davon können Namen benutzt werden, um Werte gezielt zuzuordnen. Dabei reichen die Anfangsteile der names (Ausnahme: nach einer variablen Parameterliste müssen die names vollständig angegeben werden). Mit der function <code>missing()</code> kann überprüft werden, ob für einen formalen parameter ein entsprechender aktueller parameter fehlt. <i>Aufruf:</i> <code>&lt;valuesliste&gt;</code> <code>&lt;Parametername&gt; = &lt;values&gt;</code> <i>Beispiel:</i> <code>rnorm(10, sd = 2)</code>

parameter bei Functions werden dem value nach übergeben, Soll der damit verbundene Auf-

wand vermieden werden, so kann with help der R: Gentleman+Thakar:2000 Information direkt auf Variable zugegriffen werden. Entsprechende Techniken are in [7] beschrieben.

**Spezialfall:** Functions with names der Form `xxx<-` erweitern die Zuweisungsfunktion. **Beispiel:**

```
"inc<-" <-function (x, value) x+value
x <- 10
inc(x)<- 3
x
```

In R-Zuweisungsfunktionen **muss** das value-Argument "value" heißen.

### A.23 Debugging and Profiling

A:debug

R bietet eine Reihe von Werkzeugen zur Identifizierung von Fehlern. Diese sind besonders im Zusammenhang mit Functions hilfreich. Mit `browser()` kann in einen Browser-Modus geschaltet werden. In diesem Modus sind die üblichen R-Anweisungen möglich. Daneben gibt es eine kleine Zahl von speziellen Anweisungen. Der Browser-Modus kann mit `debug()` automatisch bei Eintritt in eine function aktiviert werden. Durch den speziellen Prompt `Browser[xx]>` ist der Browser-Modus erkennbar. `debug`-Kontrolle steht. Fährt mit der Anweisungsausführung fort, falls `browser` direkt aufgerufen wurde.

`<return>` geht zur nächsten Anweisung, falls die function unter `debug`-Kontrolle steht. Fährt mit der Anweisungsausführung fort, falls `browser` direkt aufgerufen wurde.

`n` geht zur nächsten Anweisung (auch falls `browser` direkt aufgerufen wurde).

`cont` Fährt mit der Anweisungsausführung fort.

`c` Kurzform für `cont`. Fährt mit der Anweisungsausführung fort.

`where` Zeigt Aufrufverschachtelung.

`Q` Stoppt Ausführung und springt in Grundzustand zurück.

Debug-Hilfen	
<code>browser()</code>	Hält die Ausführung an und geht in den Browser-Modus. Aufruf: <code>browser()</code>
<code>recover()</code>	<code>recover()</code> zeigt eine Liste der aktuellen Aufrufe, aus der einer zur <code>browser()</code> -Inspektion gewählt werden kann. Mit <code>c</code> kehrt man aus dem <code>browser</code> zu <code>recover</code> zurück. Mit <code>Q</code> verlässt man <code>recover()</code> Aufruf: <code>recover()</code> Hinweis: Mit <code>options(error = recover)</code> kann die Fehlerbehandlung so konfiguriert werden, dass im Fehlerfall automatisch <code>browser()</code> aufgerufen wird.
<code>debug()</code>	Markiert eine function zur Debugger-Kontrolle. Bei nachfolgenden Aufrufen der function wird der Debugger aktiviert und schaltet in den Browser-Modus. Aufruf: <code>debug(&lt;function&gt;)</code>
<code>undebug()</code>	Löscht Debugger-Kontrolle für eine function. Aufruf: <code>undebug(&lt;function&gt;)</code>

(Fortsetzung)→

```
Debugging
Profiling
browser@browser|textit
debug@debug|textit
Topic
de-
bug-
ging!browser@browser
Topic
de-
bug-
ging!recover@recover
Topic
de-
bug-
ging!debug@debug
Topic
de-
bug-
ging!traceback@trace
```

<b>Debug-Hilfen</b> (Fortsetzung)	
<b>trace()</b>	Markiert eine function zur Trace-Kontrolle. Bei nachfolgenden Aufrufen der function wird der Aufruf with seinen Argumenten angezeigt. <i>Aufruf:</i> <code>trace(&lt;function&gt;)</code>
<b>untrace()</b>	Löscht Trace-Kontrolle for eine function. <i>Aufruf:</i> <code>untrace(&lt;function&gt;)</code>
<b> traceback()</b>	Im Fehlerfall innerhalb einer function wird die aktuelle Aufrufverschachtelung in einer Variablen <code>.Traceback</code> gespeichert. <code> traceback()</code> valuest diese Variable aus and zeigt den Inhalt an. <i>Aufruf:</i> <code> traceback()</code>
<b>try()</b>	Erlaubt benutzer-definierte Fehlerbehandlung. <i>Aufruf:</i> <code> traceback(&lt;Ausdruck&gt;)</code>

Um die Laufzeit in einzelnen Bereichen zu messen, bietet R ein “profiling”, das jedoch nur verfügbar is, wenn R with den entsprechenden Optionen compiliert worden is. The beim Compilieren benutzten Informationen können with `capabilities()` erfragt werden.

<b>Profiling-Hilfen</b>	
<b>system.time()</b>	Misst die Ausführungszeit einer Anweisung. Diese function is stets verfügbar. <i>Aufruf:</i> <code>system.time(&lt;expr&gt;, &lt;gcFirst&gt;)</code>
<b>Rprof()</b>	Registriert periodisch die jeweils aktiven Functions. Diese function is nur verfügbar, wenn R for “profiling” compiliert is. Mit <code>memory.profiling = TRUE</code> wird außer der Zeit auch periodisch die Speicherplatznutzung protokolliert. Diese Option is nur verfügbar, wenn R entsprechend compiliert is. <i>Aufruf:</i> <code>Rprof(filename = "Rprof.out", append = FALSE, interval = 0.02, memory.profiling = FALSE)</code>
<b>Rprofmem()</b>	Registriert Speicherplatz-Anforderungen im Anforderungsfall. Diese function is nur verfügbar, wenn R for “memory profiling” compiliert is. <i>Aufruf:</i> <code>Rprofmem(filename = "Rprofmem.out", append = FALSE, threshold = 0)</code>
<b>summaryRprof()</b>	Fasst die Ausgabe von <code>Rprof()</code> zusammen and berichtet den Zeitbedarf je function. <i>Aufruf:</i> <code>summaryRprof(filename = "Rprof.out", chunksizes = 5000, memory = c("none", "both", "tseries", "stats"), index = 2, diff = TRUE, exclude = NULL)</code>

**A.24 Control Structures**

function()

R Kontrollstrukturen	
<b>if</b>	Bedingte Ausführung <i>Aufruf:</i> <code>if (&lt;log. Ausdruck 1&gt;) &lt;Ausdruck2&gt;</code> Der logische Ausdruck 1 darf nur einen logischen value ergeben. für vektorisierten Zugriff benutze man <b>ifelse</b> . <i>Aufruf:</i> <code>if (&lt;log. Ausdruck1&gt;) &lt;Ausdruck2&gt; else &lt;Ausdruck3&gt;</code>
<b>ifelse</b>	Elementweise bedingte Ausführung <i>Aufruf:</i> <code>ifelse(&lt;log. Ausdruck1&gt;, &lt;Ausdruck2&gt;, &lt;Ausdruck3&gt;)</code> wertet den logischen Ausdruck 1 elementweise auf einen vector an, und übergibt bei wahrer Resultat den elementweisen value von Ausdruck2, sonst von Ausdruck3) <i>Beispiel:</i> <code>trimmedX &lt;- ifelse (abs (x)&lt;2, X, 2)</code>
<b>switch</b>	Auswahl aus einer Liste von Alternativen <i>Aufruf:</i> <code>switch(&lt;Ausdruck1&gt;, ...)</code> Ausdruck1 muss einen numerischen value or eine Zeichenkette ergeben. ... ist eine explizite Liste der Alternativen. <i>Beispiel:</i> <code>centre &lt;- function (x , type) { switch(type,  mean = mean(x),  median = median(x),  trimmed = mean(x, trim = .1)}</code>
<b>for</b>	Iteration (Schleife) <i>Aufruf:</i> <code>for ((name) in &lt;Ausdruck1&gt;) &lt;Ausdruck2&gt;</code>
<b>repeat</b>	Wiederholung. Muss z.B. with <b>break</b> verlassen werden. <i>Aufruf:</i> <code>repeat &lt;Ausdruck&gt;</code> <i>Beispiel:</i> <code>pars&lt;-init  repeat { res&lt;- get.resid (data, pars)  if (converged(res) ) break  pars&lt;-new.fit (data, pars) }</code>
<b>while</b>	Bedingte Wiederholung <i>Aufruf:</i> <code>while (&lt;log. Ausdruck&gt;) &lt;Ausdruck&gt;</code> <i>Beispiel:</i> <code>pars&lt;-init; res &lt;- get.resid (data, pars) while  (!converged(res)) { pars&lt;-  new.fit(data, pars) res&lt;- get.resid}</code>
<b>break</b>	verlässt die aktuelle Schleife
<b>next</b>	verlässt einen Schleifenzyklus and springt zum nächsten



**A.25 Administration and Customisation****s:A.7**

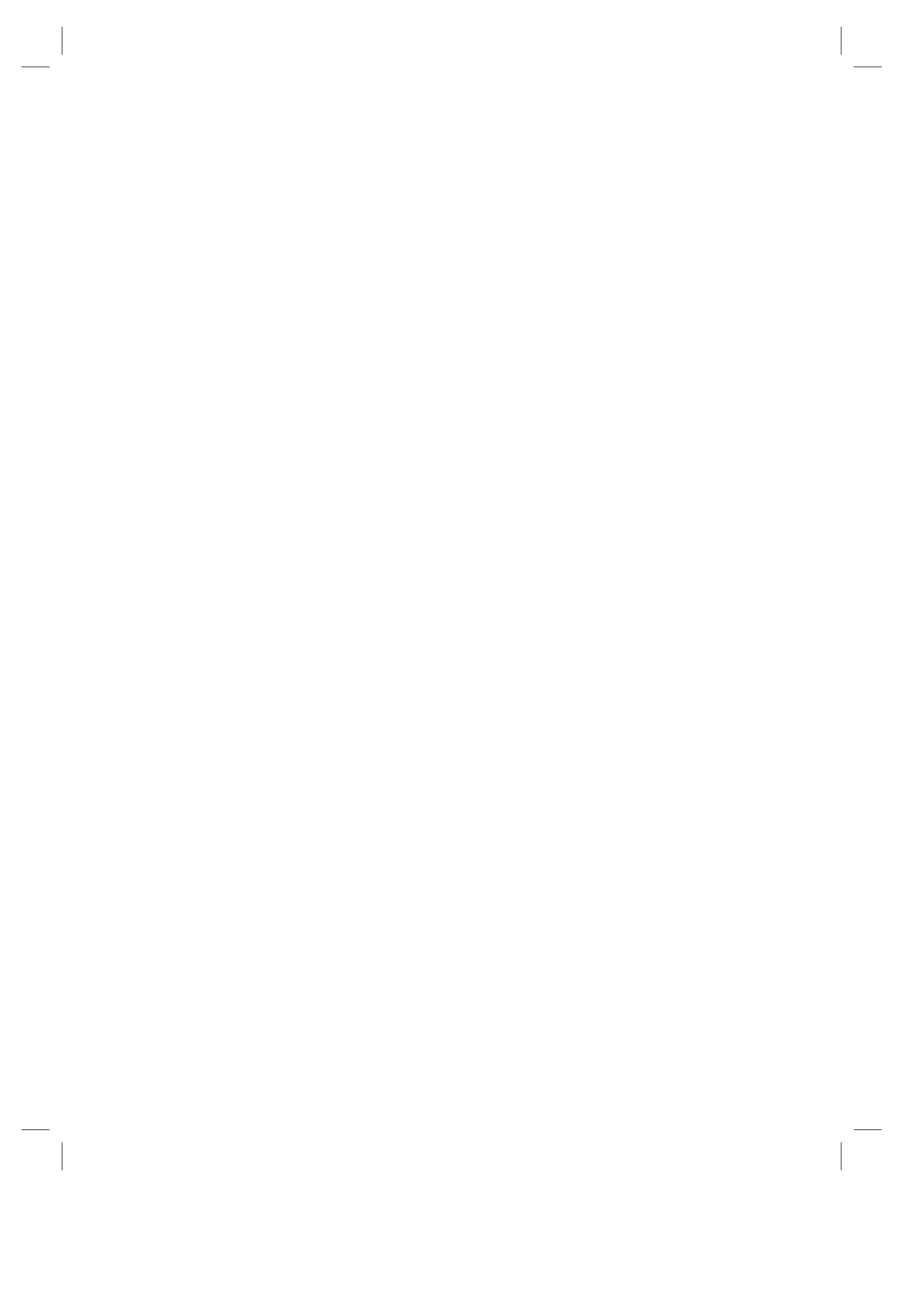
<i>objects()</i> <i>ls()</i>	Liste der aktuellen objects
<i>rm()</i>	Löscht die angegebenen objects <i>Aufruf:</i> <i>rm(&lt;Objektliste&gt;)</i>



**A.26 Input and Output to Data Streams**

s:A.8

R <i>Ein/Ausgabe</i>	
<code>write()</code>	Schreibt Daten in eine Datei. <i>Aufruf:</i> <code>write(val, file)</code> <i>Beispiel:</i> <code>write(x, file = "data")</code>
<code>source()</code>	Führt die R-Anweisungen aus der angegebenen Datei aus. <i>Aufruf:</i> <code>source("&lt;Dateiname&gt; ")</code> <i>Beispiel:</i> <code>source("cmds.R")</code>
<code>sink()</code>	Lenkt Ausgaben in die angegebene Datei. <i>Aufruf:</i> <code>sink("&lt;Dateiname&gt;")</code> <i>Beispiel:</i> <code>sink()</code> lenkt die Ausgabe wieder auf die Konsole.
<code>dump()</code>	Schreibt für ein Objekt die definierenden Kommandos. Mit <code>source()</code> kann aus der Ausgabe das Objekt regeneriert werden <i>Aufruf:</i> <code>dump(list, file = "&lt;dumpdata.R&gt;", append = FALSE)</code>



### A.27 External Data

**A.9Daten**

Zum Editieren and for die Eingabe nach Spreadsheet-Art innerhalb von R gibt es `edit()` (früherer name: `data.entry()`).

for den Austausch müssen die Datenformate zwischen allen Beteiligten abgestimmt sein. Zum Import aus Datenbanken and anderen Paketen steht eine Reihe von Bibliotheken zur Verfüzung, z.B. `stataread` for Stata, `foreign` for SAS, Minitab and SPSS, `RODBC` for SQL. Weitere Information findet sich im Manual “Data Import/Export” ([15]).

Innerhalb von R werden vorbereitete Daten üblicherweise als `data frames` bereitgestellt. Sind zusätzliche objects wie Functions or parameter nötig, so können sie gebündelt als Paket bereit gestellt werden (siehe Aufgabe A.28 (Seite A-81)).

for den Austausch zu R kann ein spezielles Austauschformat benutzt werden. Dateien in diesem Format können with `save()` generiert werden and haben konventionell die nameendung `.Rda`. Diese Dateien werden with `load()` wieder geladen.

Daten werden allgemeiner with der function `data()` geladen. Abhängig von der Endung des Dateiname der Eingabedatei verzweigt `data()` in mehreren Spezialfällen. Neben den `.Rda` are übliche Endungen for reine Daten-Eingabedateien `.tab` or `.txt`. The online-help-function `help(data)` gibt weitere Auskunft.

```
edit@edit|textit
data.entry@data.entry|textit
save@save|textit
load@load|textit
data@data|textit
data@data|textit
```

<i>Ein- Ausgabe von Daten for R</i>	
<code>save()</code>	Speichert Daten in externe Datei. <i>Aufruf:</i> <code>save(&lt;names der zu speichernden objects&gt;, file = &lt;Dateiname&gt;, ...)</code>
<code>load()</code>	Lädt Daten aus exterener Datei. <i>Aufruf:</i> <code>load(file = &lt;Dateiname&gt;, ...)</code>
<code>data()</code>	Lädt Daten. <code>data()</code> kann unterschiedliche Formate verarbeiten, wenn die Zugriffspfade and Datei-names den R-conventions folgen. <i>Aufruf:</i> <code>data(..., list = character(0), package = c(.packages(), .Autoloaded), lib.loc = .lib.loc)</code> <i>Beispiel:</i> <code>data(crimes) # lädt den Datensatz 'crimes'</code>

for den flexiblen Austausch with anderen Programmen werden Daten in der Regel als Text-Dateien bereitgestellt, nach Möglichkeit

- in Tabellenform,
- nur ASCII-Zeichen (z.B. keine Umlaute!)
- Variablen spaltenweise angeordnet
- Spalten durch Tabulator-Sprünge getrennt.
- evtl. Spaltenüberschriften in Zeile 1

```

read.table@read.table|textit
• evtl. Zeilennr. in Spalte 1.
write.table@write.table|textit
read.table@read.table|textit
Dafür wird zum Lesen die function read.table() und zum Schreiben die function write.table() benutzt.
DateTimeClasses
Datum!_see Zeitgestellt. Neben read.table() gibt es eine Reihe von Varianten, die auf andere gebräuchliche
DateTimeClassess
Zeit!_see Zeitgestellt. Datenformate abgestimmt sind. Diese sind unter help(read.table) aufgeführt.
Zeit!_see Zeitgestellt
scan@scan|textit
read.fwf@read.

```

<i>Ein- Ausgabe von Daten zum Aus- tausch</i>	
<code>read.table()</code>	<p>Liest Daten-Tabelle</p> <p>Aufruf: <code>read.table(file, header = FALSE, sep = "\t", ...)</code></p> <p>Beispiele: <code>read.table(&lt;Dateiname&gt;, header = TRUE, sep = '\t')</code>  Überschriften in Zeile 1, Zeilenr. in Spalte 1  <code>read.table(&lt;Dateiname&gt;, , header = TRUE, sep = '\t')</code>  keine Zeilenr., Überschriften in Zeile 1,</p>
<code>write.table()</code>	<p>Schreibt Daten-Tabelle</p> <p>Aufruf: <code>write.table(file, header = FALSE, sep = '\t', ...)</code></p> <p>Beispiele: <code>write.table(&lt;data frame&gt;, &lt;Dateiname&gt;, header = TRUE, sep = '\t')</code>  Überschriften in Zeile 1, Zeilenr. in Spalte 1  <code>write.table(&lt;data frame&gt;, &lt;Dateiname&gt;, header = TRUE, sep = '\t')</code>  keine Zeilenr., Überschriften in Zeile 1,</p>

Defaultmäßig konvertiert `read.table()` Daten in `factor`-Variable, falls möglich. Dieses Verhalten kann mit dem Parameter `as.is` beim Aufruf von `read.table()` modifiziert werden. Diese Modifikation ist z.B. nötig, um Datums- und Zeitangaben einzulesen, wie in dem folgenden Beispiel aus [8]:

```

# date col in all numeric format yyyy-mm-dd
df <- read.table("laketemp.txt", header = TRUE)
as.Date(as.character(df$date), "%Y-%m-%d")
# first two cols in format mm/dd/yy hh:mm:ss
# Note as.is = in read.table to force character
library("chron")
df <- read.table("oxygen.txt", header = TRUE,
as.is = 1:2)
chron(df$date, df$time)

```

for sequentielles Lesen steht `scan()` zur Verfügung. Dateien mit stellengenau fest vorgegebenem Format können mit `read.fwf()` gelesen werden.

## A.28 Libraries, Packages

s:A.9

```
library@library|textit
data@data|textit
```

Externe Information kann in (Text)-Dateien und Paketen(Packages) gespeichert sein. Bibliotheken and Pakete are dabei nach speziellen R-conventions strukturiert. “Bibliotheken” are Sammlungen von “Paketen”.

Zusätzliche Functions werden in der Regel als Pakete bereitgestellt. Pakete werden with  
`library()`

geladen. Im Paket enthaltene Datensätze are dann direkt auffindbar and werden with  
`data()`

(ohne Argument) aufgelistet.

Beispiel:

```
library(nls)
data()
data(Puromycin)
```

Pakete	
<code>library()</code>	Lädt Zusatzpaket <i>Aufruf:</i> <code>library(package, ...)</code> <i>Siehe auch</i> Abschnitt I.5.6
<code>require()</code>	Lädt Zusatzpaket; gibt Warnung bei Fehler. <i>Aufruf:</i> <code>require(package, ...)</code>
<code>detach()</code>	Gibt Zusatzpaket frei and entfernt es aus dem Suchpfad. <i>Aufruf:</i> <code>detach(&lt;name&gt;)</code>
<code>install.packages()</code>	Installiert Pakete in <code>&lt;lib&gt;</code> , lädt sie bei Bedarf aus dem Archiv CRAN <i>Aufruf:</i> <code>install.packages(pkgs, lib, CRAN =getOption("CRAN"), ...)</code>
<code>package.manager()</code>	Falls implementiert: Interface zur Verwaltung installierter Pakete. <i>Aufruf:</i> <code>package.manager()</code>
<code>package.skeleton()</code>	Erstellt das Gerüst for ein neues Paket. <i>Aufruf:</i> <code>package.skeleton(name = "&lt;anRpackage&gt;", list, ...)</code>

Detailinformation zur Erstellung von R-Paketen findet man in “Writing R Extensions” ([R:Ext](#)) ([18]).



### A.29 Linear Algebra

for die lineare Algebra are die wichtigsten Functions weitgehend standardisiert and in C-Bibliotheken wie BLAS/ATLAS and Lapack verfügbare. R benutzt diese Bibliotheken and bietet for die wichtigsten Functions einen direkten Zugang.

<i>Lineare Algebra</i>	
<code>eigen()</code>	Berechnet Eigenvalues and Eigenvektoren von reellen or komplexen Matrizen
<code>svd()</code>	Eigenwertzerlegung einer Matrix
<code>qr()</code>	QR-Zerlegung einer Matrix
<code>determinant()</code>	Determinante einer Matrix
<code>solve()</code>	Löst lineare Gleichung

Falls möglich sollten jedoch statistische Funktionen benutzt and der direkte Zugriffe auf Functions der linearen Algebra vermieden werden.



### A.30 Model Descriptions

s:A.10

Lineare statistische Modelle können durch Angabe einer Design-Matrix  $X$  spezifiziert werden und in der allgemeinen Form

$$Y = X\beta + \varepsilon$$

dargestellt werden, wobei die Matrix  $X$  jeweils genauer bestimmt werden muß.

R erlaubt es, Modelle auch dadurch zu spezifizieren, dass die Regeln angegeben werden, nach denen die Design-Matrix gebildet wird.

Operator	Syntax	Bedeutung	Beispiel
$\sim$	$Y \sim M$	$Y$ hängt von $M$ ab	$Y \sim X$ resultiert in $E(Y) = a + bX$
$+$	$M_1 + M_2$	$M_1$ und $M_2$ einschliessen	$Y \sim X + Z$ $E(Y) = a + bX + cZ$
$-$	$M_1 - M_2$	$M_1$ einschliessen, aber $M_2$ ausschliessen	$Y \sim X - 1$ $E(Y) = bX$
:	$M_1 : M_2$	Tensorprodukt, d.h. alle Kombinationen von Stufen von $M_1$ und $M_2$	
$\% \text{ in } \%$	$M_1 \% \text{ in } \% M_2$	modifiziertes Tensorprodukt	$a + b \% \text{ in } \% a$ entspricht $a + a : b$
*	$M_1 * M_2$	“gekreuzt”	$M_1 + M_2$ entspricht $M_1 + M_2 + M_1 : M_2$
/	$M_1 / M_2$	“geschachtelt”: $M_1 + M_2 \% \text{ in } \% M_1$	
$^n$	$M^n$	$M$ mit allen “Interaktionen” bis Stufe $n$	
$I()$	$I(M)$	Interpretiere $M$ . Terme in $M$ behalten ihre ursprüngliche Bedeutung; das Resultat bestimmt das Modell.	$Y \sim (1 + I(X^2))$ entspricht $E(Y) = a + bX^2$

Table A.53 Wilkinson-Rogers-Notation for lineare Modelle

ta:wrn

The Modell-Spezifikation ist auch für allgemeinere, nicht lineare Modelle möglich.

examples

$$y \sim 1 + x \quad \text{entspricht } y_i = (1 \ x_i)(\beta_1 \ \beta_2)^\top + \varepsilon$$

`update@update|textit`

$y \sim x$	Kurzschriftweise for $y \sim 1 + x$ (Konstanter Term wird implizit angenommen)
$y \sim 0 + x$	entspricht $y_i = x_i \cdot \beta + \varepsilon$
$\log(y) \sim x_1 + x_2$	entspricht $\log(y_i) = (1 \ x_{i1} \ x_{i2})(\beta_1 \ \beta_2 \ \beta_3)^\top + \varepsilon$ (Konstanter Term wird implizit angenommen)
$y \sim A$	Einweg-Varianzanalyse with Faktor A
$y \sim A + x$	Covarianzanalyse with Faktor A and Covariable x
$y \sim A * B$	Zwei-Faktor-Kreuz-Layout with Faktoren A and B
$y \sim A/B$	Zwei-Faktor hierarchisches Layout with Faktor A and Subfaktor B

Um zwischen verschiedenen Modellen ökonomisch wechseln zu können, steht die function `update()` zur Verfügung.

<b>Modell-Verwaltung</b>	
<code>formula()</code>	extrahiert Modellformel aus a Objekt
<code>terms()</code>	extrahiert Terme der Modell-Formal aus a Objekt
<code>contrasts()</code>	spezifiziert Kontraste
<code>update()</code>	Wechsel zwischen Modellen
<code>model.matrix()</code>	Generiert die Design-Matrix zu a Modell

**ToDo:**  
discuss  
`update`

Anwendungsbeispiel:

```
lm(y ~ poly(x, 4), data = experiment)
```

analysiert den Datensatz "experiment" with a linearen Modell for polynomiale Regression vom Grade 4.

<b>Standard-Analysen</b>	
<code>lm()</code>	lineares Modell Siehe auch Kapitel <a href="#">ch:02</a>
<code>glm()</code>	generalisiertes lineares Modell
<code>nls()</code>	nicht-lineare kleinste Quadrate
<code>nlm()</code>	allgemeine nicht-lineare Minimierung
<code>update()</code>	Wechsel zwischen Modellen
<code>anova()</code>	Varianz-Analyse

### A.31 Graphic Functions

s:A.11

R bietet zwei Grafik-Systeme: Das Basis-Grafiksystem von R implementiert ein Modell, dass an der Vorstellung von Stift und Papier orientiert ist. Das Lattice-Grafiksystem ist ein zusätzliches zweites Grafiksystem, dass an a Kamera/Objekt-Modell orientiert ist. Information über Lattice erhält man with `help(Lattice)` (Großbuchstabe L!), eine Übersicht über die Functions in Lattice with `library(help = lattice)`. Informationen über das Basis-Grafiksystem folgen hier.

Grafik-Functions fallen im wesentlichen in drei Gruppen:

- “high level”-Functions. Diese definieren eine neue Ausgabe.
- “low level”-Functions. Diese modifizieren eine vorhandene Ausgabe.
- Parametrisierungen. Diese modifizieren die Voreinstellungen des Grafik-Systems.

#### A.31.1 High Level Graphics

“high level”	
<code>plot()</code>	Generische Grafikfunktion
<code>pairs()</code>	paarweise Scatterplots
<code>coplot()</code>	Scatterplots, bedingt auf Covariate
<code>qqplot()</code>	Quantil-Quantil-Plot
<code>qqnorm()</code>	Gauß-Quantil-Quantil-Plot
<code>qqline()</code>	fügt eine Linie zu a Gauß-Quantil-Quantil-Plot hinzu, die durch das erste and dritte Quantil verläuft.
<code>hist()</code>	Histogramm Siehe auch Abschnitt I.3.2, Seite I-29
<code>boxplot()</code>	Box&Whisker-Plot
<code>dotplot()</code>	
<code>curve()</code>	valuest eine function or einen Ausdruck nach Bedarf aus and zeichnet eine Kurve. Beispiel: <code>curve(dnorm, from = -3, to = 3)</code>
<code>image()</code>	farbcodiertes z gegen x, y
<code>contour()</code>	Contourplot von z gegen x, y
<code>persp()</code>	3D-Fläche

#### A.31.2 Low Level Grapics

The high-level-Functions haben in der Regel einen parameter `add`. Wird beim Aufruf `add = FALSE` gesetzt, so können sie auch benutzt werden, um zu a vorhandenen Plot Elemente hinzufügen.

**Annotation**  
**Legende**  
**Beschriftung**

zu fügen. Daneben gibt es eine Reihe von low-level-Functions, die voraussetzen, dass bereits eine Plot-Umgebung geschaffen ist.

<i>"low level"</i>	
<code>points()</code>	Generische function. Markiert Punkte an angegebenen Koordinaten. Aufruf: <code>points(x, ...)</code>
<code>lines()</code>	Generische function. Verbindet Punkte an angegebenen Koordinaten. Aufruf: <code>lines(x, ...)</code>
<code>abline</code>	Fügt Linie (in mehreren Darstellungen) zum Plot hinzu. Aufruf: <code>abline(a, b, ...)</code>
<code>polygon()</code>	Fügt Polygon with spezifizierten Ecken hinzu.
<code>axis()</code>	Fügt Achsen hinzu.

Daneben hat R rudimentäre Möglichkeiten for Interaktion with Grafik.

<i>Interaktionen</i>	
<code>locator()</code>	bestimmt die Position von Mausklicks. Eine aktuelle Grafik muss definiert sein, bevor <code>locator()</code> benutzt wird. Beispiel: <code>plot(runif(19))</code> <code>locator(n = 3, type = "l")</code>

### A.31.3 Annotations and Legends

The high-level-function bieten in der Regel die Möglichkeiten, Standard-Beschriftungen durch geeignete parameter zu kontrollieren.

```
main =      Haupt-Überschrift, über dem Plot
sub =       Plot-Unterschrift
xlab =      Beschriftung der x-Achse
ylab =      Beschriftung der y-Achse
```

Beschreibungen erhält man with `help(plot.default)`.

Zur Ergänzung stehen low-level-Functions bereit.

<i>"low level"</i>	
	(Fortsetzung)→

<i>"low level"</i> (Fortsetzung)		expression@expression tex quote@bquote textit
<b>title()</b>	Setzt Überschrift, analog high-level-Parametern. <i>Aufruf:</i> <code>title(main = NULL, sub = NULL, xlab = NULL, ylab = NULL, ...)</code>	
<b>text</b>	Fügt Text an spezifizierten Koordinaten hinzu. <i>Aufruf:</i> <code>text(x, y = NULL, text, ...)</code>	
<b>legend()</b>	Fügt einen Block mit einer Legende hinzu. <i>Aufruf:</i> <code>legend(x, y = NULL, text, ...)</code>	
<b>mtext()</b>	Fügt Randbeschriftung hinzu. <i>Aufruf:</i> <code>mtext(text, side = 3, ...)</code> . The Ränder werden bezeichnet durch 1 = unten, 2 = links, 3 = oben, 4 = rechts)	

R gibt auch (eingeschränkte) Möglichkeiten zum Formelsatz. Ist der Text-parameter eine Zeichenkette, so wird sie direkt übernommen. Ist der Text-parameter ein (unausgevaluester) R-Ausdruck, so wird versucht, die mathematisch übliche Darstellung zu geben. R-Ausdrücke können with den Functions `expression()` or `bquote()` erzeugt werden.

Beispiel:

```
text(x, y, expression(paste(bquote("(",
atop(n, x), ")"),
.(p)^x, .(q)^{n-x})))
```

Ausgabe-examples erhält man with `demo(plotmath)`.

#### A.31.4 Gropic Parameters and Layout

<i>Parametrisierung</i>	
<b>par()</b>	Setzt parameter des Basis-Grafiksystems <i>Aufruf:</i> siehe <code>help(par)</code> <i>Beispiel:</i> <code>par(mfrow = c(m, n))</code> unterteilt den Grafikbereich in <i>m</i> Zeilen und <i>n</i> Spalten, die Zeile for Zeile gefüllte werden. <code>par(mfcol = c(m, n))</code> füllt den Bereich Spalte for Spalte.
<b>split.screen()</b>	Teilt den Grafik-Bereich in Teile <i>Aufruf:</i> <code>split.screen(figs, screen, erase = TRUE)</code> . Hat <i>figs</i> zwei Einträge, so werden damit die Anzahl der Zeilen and Spalten festgelegt. Ist <i>figs</i> eine Matrix, so gibt jede Zeile die Koordinaten eines Grafikbereichs in relativen Koordinaten [0...1] an. <code>split.screen()</code> kann auch geschachtelt werden.

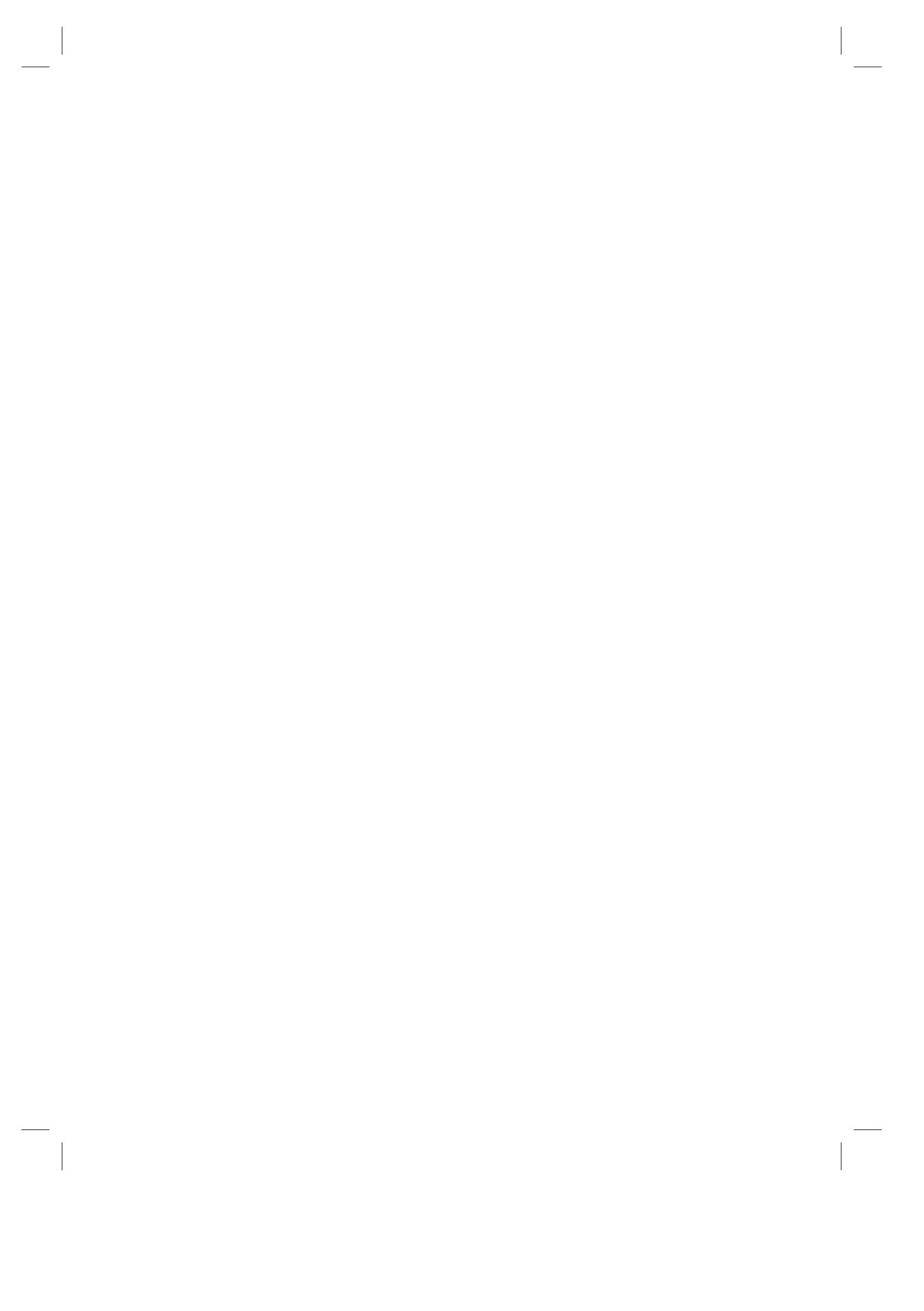
(Fortsetzung)→

<i>Parametrisierungen</i> (Fortsetzung)	
<b>screen()</b>	Wählt Grafik-Bereich for die nächste Ausgabe. <i>Aufruf:</i> <code>screen(n = cur.screen , new = TRUE).</code>
<b>layout()</b>	Unterteilt den Grafik-Bereich. Diese function is with anderen Layout-Functions nicht verträglich.

### A.32 Basic Statistical Functions

s:A.12

<i>Statistik-Functions</i>	
<i>sum()</i>	summiert Komponenten eines Vektors
<i>cumsum()</i>	bildet kumulierte Summen
<i>prod()</i>	multipliziert Komponenten eines Vektors
<i>cumprod()</i>	bildet kumulierte Produkte
<i>length()</i>	Länge eines Objekts, z.B. Vektors
<i>max()</i>	Maximum, Minimum.
<i>min()</i>	Siehe auch <i>pmax</i> , <i>pmin</i>
<i>range()</i>	Minimum and Maximum
<i>cummax()</i> <i>cummin()</i>	Kumulatives Maximum, Minimum
<i>quantile()</i>	Stichprobenquantile. for theoretische distributions: <i>qxxxx</i> , z.B. <i>qnorm</i>
<i>median()</i>	Median
<i>mean()</i>	Mittelwert auch getrimmte Mittel
<i>var()</i>	Varianz, Varianz / Covarianzmatrix
<i>sort()</i> <i>rev()</i>	Sortierung
<i>order()</i>	Sortierung nach Leit-Elemen, auch for mehrere Variable
<i>rev()</i>	Umgekehrte Sortierung
<i>rank()</i>	Stichprobenränge



**A.33 Distributions, Random Numbers, Densities...**

random\_numbers!reproduzie

s:A.13

Der Basis-Generator for uniforme random numbers wird von `Random` verwaltet. Verschiedene Basis-Generatoren stehen zur Verfügung. **for ernsthafte Simulation wird eine Lektüre der Empfehlungen von Marsaglia et al. dringend empfohlen.** (Siehe `help(.Random.seed)`). Alle nicht-uniformen Zufallsnumbersgeneratoren are vom aktuellen Basisgenerator abgeleitet. Eine Übersicht über die wichtigsten nicht-uniformen Zufallsnumbersgeneratoren, ihre Verteilungsfunktionen and ihre Quantile findet sich am Ende dieses Abschnitts.

R random numbers	
<code>.Random.seed</code>	<code>.Random.seed</code> is eine globale Variable, die den augenblicklichen Zustand des Zufallsnumbersgenerators speichert. Diese Variable kann gesichtet and with <code>set.seed()</code> wieder restauriert werden.
<code>set.seed()</code>	initialisiert den Zufallsnumbersgenerator <i>Aufruf:</i> <code>set.seed(seed, kind = NULL)</code>
<code>RngKind()</code>	<code>RngKind()</code> gibt den names des aktuellen Basisgenerators. <code>RngKind (&lt;name&gt;)</code> setzt einen Basisgenerator. <i>Aufruf:</i> <code>RngKind()</code> <code>RngKind(&lt;name&gt;)</code> <i>Beispiel:</i> <code>RngKind("Wichmann-Hill")</code> <code>RngKind("Marsaglia-Multicarry")</code> <code>RngKind("Super-Duper")</code>
<code>sample()</code>	<code>sample()</code> zieht eine Zufallsstichprobe aus den im vector <i>x</i> angegebenen valuesn, with or ohne Zurücklegen (je nach value von <code>replace</code> ).  Size is defaultmäßig die Länge von <i>x</i> .  Optional kann <code>prob</code> ein vector von probabiliyen for die values von <i>x</i> sein. <i>Aufruf:</i> <code>sample(x, size, replace = FALSE, prob)</code> <i>Beispiel:</i> Zufällige Permutation: <code>sample(x)</code>  <code>val&lt;-c("H", "T")</code> <code>prob&lt;-c(0.3, 0.7)</code> <code>sample(val, 10,</code> <code>replace = T, prob)</code>

Sollen Simulationen reproduzierbar sein, so muss der Zufallsnumbersgenerator in einen kontrollierten Zustand gesetzt sein. Ein Beispiel dafür is die folgende Anweisungsfolge: `save.seed <- .Random.seed`

```
save.seed <- .Random.seed
```

```
save.kind <- RNGkind()
```

Mit `set.seed(save.seed, save.kind)` wird dann der Zustand des Generators bei Bedarf restauriert.

The einzelnen Funktionsnames for die wichtigsten nicht-uniformen Generatoren and Functions setzen sich aus a Präfix and dem Kurznames zusammen. Allgemeiner Schlüssel: xxxx is der Kurzname

- rxxxx** erzeugt random numbers
- dxxxx** Dichte or probabilit
- pxxxx** distribution function
- qxxxx** Quantile

Beispiel:

```
x<-runif(100)      erzeugt 100 U(0,1)-verteilte Zufallsvariable
qf(0.95, 10, 2)    berechnet das 95%-Quantil der F(10,2)-distribution.
```

<i>distributions</i>	<i>Kurzname</i>	<i>parameter and Default-values</i>
Beta	<i>beta</i>	<i>shape1, shape2, ncp = 0</i>
Binomial	<i>binom</i>	<i>size, prob</i>
Cauchy	<i>cauchy</i>	<i>location = 0, scale = 1</i>
$\chi^2$	<i>chisq</i>	<i>df, ncp = 0</i>
Exponential	<i>exp</i>	<i>rate = 1</i>
F	<i>f</i>	<i>df1, df2 (ncp = 0)</i>
Gamma	<i>gamma</i>	<i>shape, scale = 1</i>
Gauß	<i>norm</i>	<i>mean = 0, sd = 1</i>
Geometrisch	<i>geom</i>	<i>prob</i>
Hypergeometrisch	<i>hyper</i>	<i>m, n, k</i>
Lognormal	<i>lnorm</i>	<i>meanlog = 0, sdlog = 1</i>
Logistisch	<i>logis</i>	<i>location = 0, scale = 1</i>
Negativ-Binomial	<i>nbinom</i>	<i>size, prob</i>
Poisson	<i>pois</i>	<i>lambda</i>
Student's t	<i>t</i>	<i>df</i>
Tukey Studentised Range	<i>tukey</i>	
Uniform	<i>unif</i>	<i>min = 0, max = 1</i>
Wilcoxon Signed Rank	<i>signrank</i>	<i>n</i>
Wilcoxon Rank Sum	<i>wilcox</i>	<i>m, n</i>
Weibull	<i>weibull</i>	<i>shape, scale = 1</i>

### A.34 Programming on the Language

The Sprachausdrücke von R are genau so objects wie Daten or Functions. Wie diese können sie gelesen or verändert werden.

<i>Umwandlungen</i>	
<code>parse()</code>	Wandelt Eingabe in eine Liste von R-Ausdrücken um. <code>parse</code> führt den Parse-Schritt durch, <code>valuest</code> die Ausdrücke aber nicht aus.
<code>deparse()</code>	Wandelt einen R-Ausdruck in interner Darstellung in eine Zeichendarstellung um.
<code>expression()</code>	erzeugt einen R-Ausdruck in interner Darstellung. <i>Beispiel:</i> <code>integrate &lt;- expression(integral(fun, lims))</code> <i>Siehe auch</i> <a href="#">I.3.1: Mathematischer Formelsatz in Plot-Beschriftungen</a>
<code>substitute()</code>	R-Ausdrücke with Auswertung aller definierten Terme.
<code>bquote()</code>	R-Ausdrücke with selektiver Auswertung. Terme in <code>.()</code> werden ausgewertet. <i>Beispiele:</i> <code>n&lt;-10; bquote( n^2 == .(n*n))</code>

<i>Auswertung</i>	
<code>eval()</code>	valuest einen Ausdruck aus.

```
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rnw/S0A1.Rnw.tex,v $
$Revision: 1.1 $
$Date: 2008/02/14 18:48:40 $
$name: $
$Author: j40 $
```



## References

- [1] Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The New S Language*. Chapman & Hall, London, 1988.
- [2] John M. Chambers. *Programming with Data*. Springer, New York, 1998. ISBN 0-387-98503-4.
- [3] John M. Chambers and Trevor J. Hastie. *Statistical Models in S*. Chapman & Hall, London, 1992.
- [4] William S. Cleveland. *Visualizing Data*. AT&T Bell Laboratories, Murray Hill, 1993.
- [5] George W. Furnas and Andreas Buja. Prosection views: dimensional inference through sections and projections. *J. Comput. Graph. Statist.*, 3(4):323–385, 1994.
- [6] Peter Gänßler and Winfried Stute. *Wahrscheinlichkeitstheorie*. Springer, 1977.
- [7] Robert Gentleman and Ross Ihaka. Lexical scope and statistical computing. *Journal of Computational and Graphical Statistics*, 9:491–508, 2000.
- [8] Gabor Grothendieck and Thomas Petzoldt. R help desk: Date and time classes in R. *R News*, 4(1):29–32, June 2004.
- [9] Alfred Inselberg, Tuval Chomut, and Mordechai Reif. Convexity algorithms in parallel coordinates. *J. Assoc. Comput. Mach.*, 34(4):765–801, 1987.
- [10] N.L. Johnson and S. Kotz. *Discrete Distributions*. Wiley, New York, 1970.
- [11] Bent Jørgensen. *The Theory of Linear Models*. Chapman & Hall, New York-London, 1993.
- [12] R. G. Miller. *Simultaneous Statistical Inference*. Springer, New York, 1981.
- [13] Paul Murrell. *R Graphics*. Chapman & Hall/CRC, Boca Raton, Fla. [u.a.], 2006.
- [14] R Development Core Team. An introduction to R. Technical report, R Project, 2007.
- [15] R Development Core Team. R data import/export. Technical report, R Project, 2007.
- [16] R Development Core Team. The R language definition. Technical report, R Project, 2007.
- [17] R Development Core Team. The R reference index. Technical report, R Project, 2007.
- [18] R Development Core Team. Writing R extensions. Technical report, R Project, 2008.
- [19] C. Radhakrishna Rao. *Linear Statistical Inference and Its Applications*. Wiley, 2 edition, 1973.
- [20] G. M. Reaven and R. G. Miller. An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia*, 16:17–24, 1979.
- [21] Günther Sawitzki. Numerical reliability of data analysis systems. *Computational Statistics & Data Analysis*, 18(2):269–286, 1994.
- [22] Günther Sawitzki. Report on the numerical reliability of data analysis systems. *Computational Statistics & Data Analysis*, 18(2):289 – 301, 1994.
- [23] Günther Sawitzki. Quality control and early diagnostics for cDNA microarrays. *R News*, 2(1):6–10, March 2002.
- [24] William N. Venables and Brian D. Ripley. *S Programming*. Springer, 2000. ISBN 0-387-98966-8.
- [25] William N. Venables and Brian D. Ripley. *Modern Applied Statistics with S*. Springer, Heidelberg, 4 edition, 2002.



---

## Functions by Topic

---

Topic **aplot**  
  **coplot**, 4-16

Topic **debugging**  
  **browser**, A-71  
  **debug**, A-71  
  **recover**, A-71  
  **traceback**, A-71

Topic **distribution**  
  **qqnorm**, 3-7  
  **Uniform**, 1-4, C-1

Topic **hplot**  
  **coplot**, 4-16  
  **pairs**, 4-7  
  **qqnorm**, 3-7

Topic **htest**  
  **t.test**, 3-12  
  **wilcox.test**, 3-16

Topic **loess**  
  **loess**, 2-37

Topic **models**  
  **anova**, 2-21

Topic **regression**  
  **anova**, 2-21  
  **lm**, 2-11

Topic **smooth**  
  **loess**, 2-37

---

## Function and Variable Index

---

.Random.seed, 1-5, C-2  
[, B-2

aa1 itfun (*alias1*), B-1  
abbreviate, A-65  
abline, 1-14  
add1, 2-22  
aggregate, A-64  
anova, 2-13, 2-21, 2-30, A-86  
anova.lm, 2-14  
aov, 2-11, 2-12, 2-14, 2-22  
aperm, A-66  
apply, 1-25, A-64  
apropos, A-49, A-50  
args, A-49  
array, A-61, A-63  
as.data.frame, 2-11, 2-37  
attach, A-62  
attr, 2-42  
attributes, A-57  
axis, A-88

barchart, 4-4  
barplot, 4-4  
boxcox, 2-35  
boxplot, 1-37, 4-4, A-87  
bquote, 1-26, A-89, A-95  
browser, A-71  
bwplot, 4-4  
by, A-64

c, 1-9  
casefold, A-65  
cbind, A-61, A-64  
chartr, A-65  
chisq.test, 1-31  
citation, 1-54  
class, 2-12, 2-42, A-57  
cloud, 4-4, 4-12  
co.intervals (*coplot*), 4-16

coef, 2-14, 2-43  
coefficients, 2-22  
confint, 2-14, 2-44  
contour, 4-1, 4-2, A-87  
contourplot, 4-4  
contrasts, A-86  
coplot, 4-16, A-87  
cummax, A-91  
cummin, A-91  
cumprod, A-91  
cumsum, A-91  
curve, 1-23, A-87

data, 1-53, 2-40, A-59, A-79, A-81  
data.entry, A-79  
data.frame, A-62  
data.matrix, 4-8  
debug, A-71  
demo, A-49  
density, 1-17  
densityplot, 4-4  
deparse, A-95  
detach, A-62, A-81  
determinant, A-83  
dim, A-61, A-63  
dimnames, A-63  
dir, A-59  
dotchartt, 4-4  
dotplot, 4-4, A-87  
drop1, 2-22  
dump, A-77  
dunif (*Uniform*), 1-4, C-1  
duplicated, A-65

edit, A-79  
effects, 2-13, 2-14, 2-22, 2-43  
eigen, A-83  
environment, 1-52, A-50  
eval, 1-51, A-95  
example, A-49

**expand.grid**, A-65  
**expression**, 1-14, A-89, A-95

**factor**, 2-5, 4-18, A-61  
**find**, A-50  
**fitted**, 2-14, 2-44  
**fitted.values**, 2-22  
**formula**, 2-12, A-86  
**fun1**, B-1  
**function**, 4-17

**getwd**, A-59  
**glht**, 2-30  
**glm**, 2-14, A-86

**help**, A-49, A-59  
**help.search**, A-49  
**hist**, 1-17, 1-20, 4-4, A-87  
**histogram**, 4-4

**identify**, 4-37  
**image**, 4-1, 4-2, 4-4, 4-46, A-87  
**influence**, 2-44  
**inherits**, 2-42  
**install.packages**, 1-52, A-81  
**integrate**, 4-32

**kruskal.test**, 3-18

**lapply**, A-64  
**lattice.options**, A-59  
**lda**, 4-25  
**leaps**, 4-41  
**legend**, 1-46, A-89  
**length**, 1-14, A-50, A-57, A-91  
**levels**, A-61  
**library**, A-59, A-81  
**lines**, A-88  
**list**, A-62  
**lm**, 2-5, 2-11, 2-22, 2-43, 4-38, 4-45, A-86  
**lm.fit**, 2-12, 2-14  
**lm.influence**, 2-14  
**lm.wfit**, 2-14  
**load**, 2-40, A-79  
**locator**, A-88  
**loess**, 2-37  
**loess.control**, 2-38  
**lowess**, 2-38

**ls**, 1-52, A-50, A-59, A-75  
**ls.str**, A-50

**match**, A-65  
**matrix**, 4-18, A-63  
**max**, A-91  
**mean**, 1-36, A-91  
**median**, A-91  
**merge**, A-65  
**methods**, 2-43, A-59  
**min**, A-91  
**missing**, A-69  
**mode**, 2-42, A-50, A-53, A-57  
**model.frame**, 3-13, 3-17  
**model.matrix**, 2-12, 2-17, 2-44, A-86  
**model.matrix.default**, 2-12  
**model.offset**, 2-12  
**mtext**, 1-48, 4-18, A-89  
**mvr**, 4-45

**NA**, 3-8  
**na.exclude**, 2-12  
**na.fail**, 2-12  
**na.omit**, 2-12  
**names**, A-57  
**ncol**, A-63  
**nlm**, A-86  
**nls**, A-86  
**nrow**, A-63

**objects**, A-50, A-75  
**offset**, 2-12  
**options**, 2-12, A-59  
**order**, A-91  
**ordered**, A-61  
**outer**, 1-25, A-64

**package.manager**, A-81  
**package.skeleton**, 1-53, 1-54, A-81  
**pairs**, 4-4, 4-7, 4-7, 4-11, 4-18, 4-35, A-87  
**panel.smooth**, 4-18  
**par**, 4-18, A-59  
**parallel**, 4-4  
**parse**, 1-51, A-95  
**persp**, 4-1, 4-2, 4-4, A-87  
**plot**, 1-14, 1-15, 2-31, 2-42, 4-4, 4-23, 4-37, A-55, A-87  
**pmatch**, A-65

`points`, 4-18, A-88

`polygon`, A-88

`power.prop.test`, 3-26

`power.t.test`, 3-23

`ppoints`, 3-8

`prcomp`, 4-29, 4-42

`predict`, 2-14, 2-26, 2-44

`predict.lm`, 2-13, 2-14, 2-26, 4-45

`predict.loess`, 2-38

`print`, 1-51, 1-52, 2-42, 4-2, 4-12, A-55

`print.anova` (*anova*), 2-21

`print.lm` (*lm*), 2-11

`prod`, A-91

`prop.test`, 3-14, 3-25

`psignrank`, 3-18

`punif` (*Uniform*), 1-4, C-1

`pwilcox`, 3-18

`q`, 1-3

`qq`, 4-4

`qqline` (*qqnorm*), 3-7, A-87

`qqmath`, 4-4

`qqnorm`, 3-6, 3-7, 4-4, A-87

`qqplot`, 3-6, 4-4, A-87

`qqplot` (*qqnorm*), 3-7

`qr`, A-83

`quantile`, 1-37, A-91

`qunif` (*Uniform*), 1-4, C-1

`range`, 4-18, A-91

`rank`, A-91

`rbind`, A-61, A-64

`read.fwf`, A-80

`read.table`, A-80

`recover`, A-71

`regsubsets`, 4-40

`rep`, 1-9

`replicate`, A-64

`require`, A-81

`reshape`, A-65

`residuals`, 2-14, 2-22, 2-43

`rev`, A-91

`rm`, A-75

`RngKind`, A-93

`rnorm`, 1-5, C-2

`Rprof`, A-72

`Rprofmem`, A-72

`rug`, 1-17

`runif`, 1-4

`runif` (*Uniform*), 1-4, C-1

`sample`, A-93

`sapply`, A-64

`save`, 1-54, 2-40, A-79

`scan`, A-80

`screen`, A-90

`sd`, 1-36

`search`, 1-52, A-49, A-50, A-59

`searchpaths`, A-50

`seq`, 1-9, A-65

`set.seed`, A-93

`sink`, A-77

`solve`, A-83

some fun, B-1

`sort`, 1-14, A-91

`source`, 1-52, 1-53, A-77

`split`, A-64, A-66

`split.screen`, A-89

`splom`, 4-4

`stack`, 2-40

`stdres`, 2-43

`storage.mode`, 2-42, A-57

`str`, A-57

`stripchart`, 4-4

`stripplot`, 4-4, 4-24

`structure`, A-55, A-57

`studres`, 2-44

`substitute`, A-95

`substring`, A-65

`sum`, A-91

`summary`, 1-37, 2-22, A-55

`summary.lm`, 2-14

`summaryRprof`, A-72

`svd`, A-83

`Sweave`, 1-52

`sys.parent`, A-50

`system`, A-59

`system.time`, A-72

`t`, A-66

`t.test`, 3-12, 3-18

`table`, 1-20, A-64, A-65

`tapply`, A-61, A-64

`terms`, 2-13, A-86

`title`, 4-18, A-89

`tolower`, A-65

`toupper`, A-65

`trace`, A-72

`traceback`, *A-72*  
`trellis.par.set`, *A-59*  
`try`, *A-72*  
`ts.intersect`, *2-13*  
`typedef`, *A-53*  
`typeof`, *2-42, A-50, A-57*

`unclass`, *2-42*  
`undebug`, *A-71*  
`Uniform`, **1-4, C-1**  
`unique`, *A-65*  
`unsplit`, *A-66*  
`untrace`, *A-72*  
`update`, *A-86*  
`update.packages`, *1-53*  
`UseMethod`, *2-42, 2-43*

`var`, *1-36, A-91*  
`vcov`, *2-14, 2-44*

`wilcox.exact`, *3-18*  
`wilcox.test`, *3-15, 3-16*  
`wilcox_test`, *3-15*  
`wireframe`, *4-4*  
`write`, *A-77*  
`write.table`, *A-80*

`xyplot`, *4-4*

---

# Index

---

*PP*-Plot, **1-40**

*QQ*-Plot, **1-40**

FixMe

  A.34: Datenrahmen vs data.frames, A-62

ToDo

  A.34: 2/3, 4-31

  A.34: Add reference to lattice, 1-47

  A.34: Check revise Tukey-Anscombe, 2-19

  A.34: Comment on trellis, 4-21

  A.34: Does memory help to increase the bound?, 4-14

  A.34: Qualitative features, 3-29

  A.34: Rao translation, 1-31

  A.34: Ref to curse of dimension, 4-22

  A.34: Remark on small setosa, 4-25

  A.34: Scheffe: fix lower/upper plotting bound, 2-28

  A.34: Simulation, Power, 3-20

  A.34: Variance bias dilemma, 4-22

  A.34: add color, 4-25

  A.34: add label for examples, 1-27

  A.34: add legend, control colour, 4-15

  A.34: add mag, 4-22

  A.34: add merge data frames Harrell 4.2.5ff, A-65

  A.34: add more regression graphics, 2-1

  A.34: add power function, 3-15

  A.34: add power simulation - see Rnw.tex source, 3-16

  A.34: add references, 1-40

  A.34: adjust gray formats in tables, 3-27

  A.34: adjust scale, orientation, 4-31

  A.34: apply and related, 1-24

  A.34: better exercise, 1-43

  A.34: ch01: add examples f. outer, 1-25

  A.34: ch02: Add example, 2-40

  A.34: ch03: exercise: power, 3-10

  A.34: ch03: standardize scale, 3-10

  A.34: ch03: work out, 3-10

  A.34: check contrasts, 2-22

  A.34: check iris inverse ratio, 4-26

  A.34: check pearson, 1-30

  A.34: check plot, 2-27

  A.34: clarify: information <-> assumptions, 3-15

  A.34: comment ch.4 experimental, 12

  A.34: comment on literature, 12

  A.34: consider simple exercises for random numbers – Antony, 1-8

  A.34: copy function syntax, 1-48

  A.34: discuss alpha channel, 4-3

  A.34: discuss choice of models - model dependent estimation, 2-22

  A.34: discuss missing values - IntroR 2.5, 1-1

  A.34: discuss update, A-86

  A.34: examples of help files included, 11

  A.34: extend reshape data frames Harrell 4.2.7, A-65

  A.34: formal variance bias, 4-27

  A.34: formats: add frame. This may be long. longtable environment?, B-8

  A.34: formats: avoid page break, B-5

  A.34: formats: show text in tables ragged right. Add hyphenation, B-5

  A.34: histogram as density estimator; choice of cells, 1-32

  A.34: improve Scheffe, 2-26

  A.34: introduce noncentral t in ch. 2, 3-21

  A.34: mention unbiased, 2-7

  A.34: more precise exercise, 1-50

  A.34: move to cbind etc, A-66

  A.34: or change plot coordinates?, 4-19

  A.34: reference to kernel density estimators, 1-32

  A.34: remove line caps - bar line is too long, 1-22

  A.34: search path, 1-53

## Index

A-105

- A.34: std error, t-test, pointwise confidence, 2-18
- A.34: studentisierte Residuen definieren, 2-18
- A.34: studentized residuals, pointwise confidence intervals, 2-18
- A.34: superassignment introR 10.5, A-69
- A.34: supply better support for legend, colour key, 4-11
- A.34: syntax: stack, unstack, split, cut, 2-40
- A.34: this is first theorem. make numeration more logical, 1-22
- A.34: use attach, 4-27
- A.34: work out as example, 1-26
  
- aa2 itd (alias2)**, B-1
- Added-Variable-Plots, **4-28**
- Annotation, A-88
  
- Bandbreite, **1-12**
- bedingt, **4-16**
- Beschriftung, A-88
- Bindung, 3-15
- Bootstrap, **3-10**
- Box-Cox-Transformation, **2-35**
- brushing, **4-6**
  
- Coplot, **4-16**
- curse of dimension, **4-34**
  
- Datenstrukturen, 1-19, A-61
- DateTimeClasses, A-80
- Datum
  - see DateTimeClasses, A-80
- Debugging, 1-51, A-71
- Design-Matrix, **2-2**, **2-15**
  
- entry, **B-9**
- Erwartungswert, **1-35**
- exakter Test, 3-15
  
- Faktor, **2-4**
  - Stufen, 2-5
- Fit, **2-6**
- function, **1-47**, A-69–A-73
  - polymorphic, *siehe* polymorphic
  
- Gauß-Markov-Schätzer, **2-5**
- Güte, **1-33**
  
- Hut-Matrix, **2-7**
  
- Kleinste-Quadrate-Schätzer, **2-5**
- Kontrast, **2-22**, **2-28**
- ks.test**, 1-29
  
- Legende, A-88
- linking, **4-6**
- locfit**, 4-6
- loglin**, 1-31
  
- Marginalverteilung, **4-7**
- MASS**, 2-43, 2-44
- Modell
  - einfaches lineares, **2-8**
    - lineares, 2-2
  - Modelfunktion, **2-2**
  - mva**, 4-29
  
- parameter
  - default, 1-3
  - plot, 1-6
  - plotmath, 1-14
  - polymorph, 1-52, 2-42, 2-43, A-55
  - polymorphic, 1-3
  - probability plot, **1-40**
  - Profiling, A-71
  - projection pursuit, **4-14**
  
- Quantilplot, **1-40**
  
- random numbers, 1-4
  - Pseudo-, 1-8
  - reproduzierbare, A-93
- Regression
  - lineare, 2-2
  - Regressor, **2-2**
  - Residuum, **2-8**
  - Respons, **2-2**
  
- Scatterplot-Matrix, **4-7**
- Serienplot, **1-6**
- Shift-Familie, **3-4**
- Skala

kategorial, 2-5  
ordinal, 2-5  
Skalen-Shiftfamile, **3-5**  
smoothing, **1-12**  
Standardabweichung, **1-35**  
Stichproben  
wiederholte, **1-32**  
Stichprobenvarianz, **1-36**  
stochastisch kleiner, **3-4**  
Streuungszerlegung, **2-19**  
substitute, **B-8**

Test  
 $\chi^2$ , 1-31  
exakt, 3-15  
Kolmogorov-Smirnov, 1-29  
Median-, 1-30  
Monte-Carlo, **1-25**  
t, 3-12  
Wilcoxon, 3-15

used, B-9

Varianz, **1-35**  
residuelle, **2-8**  
Varianzanalyse, **2-19**  
Variationskoeffizient, **3-26**

Wilkinson-Rogers-Notation, **2-4**

Zeit  
see DateTimeClasses, A-80

---

## CHAPTER B

S formats. Remove for public version

---

```
aa1 it-
fun@aa1
itfun
(alias1)
aa2 itd@aa2
itd
(alias2)
some_ fun
fun1
```

```
expl:fo1
expl:fo2
```

**Example B.1** Nur *expl*

**Example B.1** Nach dem Schunk

Vor dem Schunk

```
Schunk:fo1
```

expl:fo1 **expl:fo1** **expl:fo2** **B.1.** Schunk:fo1 **B.1.**

### B.1 Test R function index

```
aa1 itfun()
aa2 itd
```

---

T<sub>E</sub>X Example Start: exa2

```
TEX Example Input: exa2
```

```
\ixr{fun1}
```

```
TEX Example Output: exa2
```

```
exa2
```

T<sub>E</sub>X Example End: exa2

---

T<sub>E</sub>X Example Start: exa3

```
TEX Example Input: exa3
```

B-2

\ixr[fun2]{fun3}

T<sub>E</sub>X Example Output: exa3

exa3

fun2]fun3 \_\_\_\_\_  
T<sub>E</sub>X Example End: exa3

---

## B.2 R styles

Sinput

---

T<sub>E</sub>X Example Start: exa4

T<sub>E</sub>X Example Input: exa4

```
\begin{Sinput}
>plot(x) # "1+1"
\end{Sinput}
```

T<sub>E</sub>X Example Output: exa4

exa4

T<sub>E</sub>X Example End: exa4

---

Soutput

---

T<sub>E</sub>X Example Start: exa5

T<sub>E</sub>X Example Input: exa5

```
\begin{Soutput}
abc <- "1+1"
\end{Soutput}
```

---

TeX Example Output: exa5

---

exa5

Output

---

TeX Example End: exa5

---

Sinput and Soutput in an Schunk

---

TeX Example Start: exa6

---

TeX Example Input: exa6

---

```
\begin{Schunk}
\begin{Sinput}
>plot(x)# "1+1"
\end{Sinput}
\begin{Soutput}
abc <- "1+1"
\end{Soutput}
\end{Schunk}
```

---

TeX Example Output: exa6

---

exa6

Input

plot(x)# "1+1"

Output

---

TeX Example End: exa6

---

irin

---

TeX Example Start: exa7

---

TeX Example Input: exa7

---

B-4

```
\begin{irin}
abc$d <- "1+1" #R input code block
\end{irin}
```

TEX Example Output: exa7

---

exa7 abc\$d <- "1+1" #R input code block

---

TEX Example End: exa7

---

irout

---

TEX Example Start: exa8

TEX Example Input: exa8

```
\begin{irout}
abc <- "1+1" #R output code block
\end{irout}
```

TEX Example Output: exa8

---

exa8 abc <- "1+1" #R output code block

---

TEX Example End: exa8

---

Scode

---

TEX Example Start: exa9

TEX Example Input: exa9

```
\begin{Scode}
abc <- "1+1" # R output code block in Scode environment
\end{Scode}
```

---

TeX Example Output: exa9

---

**exa9** `abc <- "1+1" # R output code block in Scode environment`

---

TeX Example End: exa9

---



---

TeX Example Start: exa10

---

TeX Example Input: exa10

---

A sample \ircode{abc <- "1+1"} inline code

---

TeX Example Output: exa10

---

**exa10** A sample `abc <- "1+1"` inline code  
TeX Example End: exa10

---

### B.3 Tables

Test:

---

TeX Example Start: exa11

---

TeX Example Input: exa11

---

```
\begin{aufgabe}{1}
&abc\\
de&fg\\
\hline
\end{aufgabe}
```

---

**ToDo:**

formats:  
show text  
in tables  
ragged  
right. Add  
hyphenati-  
on

**ToDo:**

formats:  
avoid page  
break

TeX Example Output: exa11

---

B-6

<b>Aufgabe B.1</b>	
	abc
de	fg

exa11

---

T<sub>E</sub>X Example End: exa11

---



---

T<sub>E</sub>X Example Start: exa12

T<sub>E</sub>X Example Input: exa12

```
\begin{tabthree}{Verteilungen}{Kurzname}{Defaults}
Beta & beta & \rcode{shape1, , shape2, , ncp=0} \\ \hline
Binomial & binom & \rcode{size, prob} \\ \hline
Cauchy & cauchy & \rcode{location = 0, scale = 1} \\ \hline
\$chi^2\$ & chisq & \rcode{df, ncp=0} \\ \hline
Exponential & exp & \rcode{rate = 1} \\ \hline
\$F\$ & \$f\$ & \rcode{df1, df2, ncp = 0} \\ \hline
Gamma & gamma & \rcode{shape, scale = 1} \\ \hline
Gauß & norm & \rcode{mean = 0, sd = 1} \\ \hline
Geometrisch & geom & \rcode{prob} \\ \hline
Hypergeometrisch & hyper & \rcode{m, n, k} \\ \hline
Lognormal & lnorm & \rcode{meanlog = 0, sdlog = 1} \\ \hline
Logistisch & logis & \rcode{location = 0, scale = 1} \\ \hline
Negativ-Binomial & nbinom & \rcode{size, prob} \\ \hline
Poisson & pois & \rcode{lambda} \\ \hline
Student's t & t & \rcode{df} \\ \hline
Tukey Studentised Range & tukey & \\ \hline
Uniform & unif & \rcode{min = 0, max = 1} \\ \hline
Wilcoxon Signed Rank & signrank & \rcode{n} \\ \hline
Wilcoxon Rank Sum & wilcox & \rcode{m, n} \\ \hline
Weibull & weibull & \rcode{shape, scale = 1}
\\
\end{tabthree}
```

---

T<sub>E</sub>X Example Output: exa12

---

<i>Verteilungen</i>	<i>Kurzname</i>	<i>Defaults</i>
Beta	beta	<i>shape1, shape2, ncp=0</i>

(cont.)→

<i>Verteilungen</i>	<i>Kurzname</i>	<i>Defaults</i>
Binomial	binom	<i>size, prob</i>
Cauchy	cauchy	<i>location = 0, scale = 1</i>
$\chi^2$	chisq	<i>df, ncp=0</i>
Exponential	exp	<i>rate = 1</i>
$F$	<i>f</i>	<i>df1, df2, ncp = 0</i>
Gamma	gamma	<i>shape, scale = 1</i>
Gauß	norm	<i>mean = 0, sd = 1</i>
Geometrisch	geom	<i>prob</i>
Hypergeometrisch	hyper	<i>m, n, k</i>
Lognormal	lnorm	<i>meanlog = 0, sdlog = 1</i>
Logistisch	logis	<i>location = 0, scale = 1</i>
Negativ-Binomial	nbinom	<i>size, prob</i>
Poisson	pois	<i>lambda</i>
Student's t	<i>t</i>	<i>df</i>
Tukey Studentised Range	tukey	
Uniform	unif	<i>min = 0, max = 1</i>
Wilcoxon Signed Rank	signrank	<i>n</i>
Wilcoxon Rank Sum	wilcox	<i>m, n</i>
Weibull	weibull	<i>shape, scale = 1</i>

exa12

---

TeX Example End: exa12

---



---



---

TeX Example Start: exa13

TeX Example Input: exa13

```
%\noindent\begin{longtable}{|p{.98\linewidth}|}
\begin{Help}
\begin{Helpinput}
help(lm)
\end{Helpinput}
\todo{formats: add frame. This may be long. longtable environment?}
\begin{Helpoutput}{help(lm)}{lm} \hfill package:base \hfill

```

R Documentation\\Fitting L

```
substitute\textbf{... see help test file}
```

```
\end{Helpoutput}
\end{Help}
%\end{longtable}
```

---

T<sub>E</sub>X Example Output: exa13

---

**ToDo:**  
formats:  
add frame.  
This may  
be **exa13**,  
longtable  
environment?

help(lm) help(lm)lm  
Fitting Linear Models

package:base

R Documentation

---

T<sub>E</sub>X Example End: exa13

---



---

T<sub>E</sub>X Example Start: exa14

---

T<sub>E</sub>X Example Input: exa14

---

An index \itd[{substitute}]{entry} definition with substitute.\\"

---

T<sub>E</sub>X Example Output: exa14

---

An index *entry* definition with substitute.

---

T<sub>E</sub>X Example End: exa14

---



---

T<sub>E</sub>X Example Start: exa15

---

T<sub>E</sub>X Example Input: exa15

---

An index \itd{entry} definition without substitute.\\"

---

T<sub>E</sub>X Example Output: exa15

---

entry|textbf  
used

---

An index *entry* definition without substitute.

exa15

---

TeX Example End: exa15

---

---

---

TeX Example Start: exa16

TeX Example Input: exa16

A \itx{used} index entry\\

TeX Example Output: exa16

A *used* index entry

exa16

---

TeX Example End: exa16

---



---

## CHAPTER C

# S help — remove for public version

---

```
Uniform@Uniform|textbf
dunif@dunif
  (Uni-
   form)
punif@punif
  (Uni-
   form)
qunif@qunif
  (Uni-
   form)
runif@runif
  (Uni-
   form)
Topic
  dis-
  tri-
  bu-
  tion!Uniform@Uniform
```

### C.1 Ch 01

---

---

#### help(runif)

---

**Uniform**

*The Uniform Distribution*

---

#### Description

These functions provide information about the uniform distribution on the interval from **min** to **max**. **dunif** gives the density, **punif** gives the distribution function **qunif** gives the quantile function and **runif** generates random deviates.

#### Usage

```
dunif(x, min=0, max=1, log = FALSE)
punif(q, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
runif(n, min=0, max=1)
```

#### Arguments

<b>x,q</b>	vector of quantiles.
<b>p</b>	vector of probabilities.
<b>n</b>	number of observations. If <b>length(n) &gt; 1</b> , the length is taken to be the number required.
<b>min,max</b>	lower and upper limits of the distribution.
<b>log, log.p</b>	logical; if TRUE, probabilities p are given as log(p).
<b>lower.tail</b>	logical; if TRUE (default), probabilities are $P[X \leq x]$ , otherwise, $P[X > x]$ .

```
.Random.seed@.Random.seed|textit  
rnorm@rnorm|textit
```

If `min` or `max` are not specified they assume the default values of `0` and `1` respectively.

The uniform distribution has density

$$f(x) = \frac{1}{max - min}$$

for  $min \leq x \leq max$ .

For the case of  $u := min == max$ , the limit case of  $X \equiv u$  is assumed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

.Random.seed about random number generation, `rnorm`, etc for other distributions.

### Examples

```
u <- runif(20)

## The following relations always hold :
punif(u) == u
dunif(u) == 1

var(runif(10000))#- ~ = 1/12 = .08333
```

---

---

## CHAPTER D

# To Do

---

### D.1 For release

Fix colour error at help page(?)  
Fix runoff text in too long input lines: ex:1.3  
Check/redo image sizes  
Fix >  $\beta$  in help(loess) family

### D.2 misc

discuss name extension  
in Sframe: remove vertical space after Beispiel.  
summar f. typeof  
add margin.table, addmargins Verzani p. 72-73  
add stripchart Verzani p. 79 add jitter Verzani p.90  
Check ‘ ToDo’ entries in index!!!  
Fix refs at exercises  
Fine tune graphics size  
Add bibliography by chapter  
Center email/URL on title page  
Grid/Lattice Check lattice vs par usage  
Classes?  
Check libraries/packages  
Fix Bibliography to german header & special words  
Check missing fonts: grep ‘LaTeX Font Info’ \*.log

```
$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/chapters/Rintro_main.tex,v $
$Revision: 1.9 $
$Date: 2008/02/12 10:05:02 $
$Name:  $
$Author: j40 $

$Source: /u/math/j40/cvsroot/lectures/src/SIntro/Rintro/Rintro_temp.tex,v $
$Revision: 1.2 $
$Date: 2008/02/12 10:05:01 $
$Name:  $
$Author: j40 $
```