

Intro Datenanalyse

Jan-Philipp Kolb

3 und 4 Mai 2017

Contents

Warum R nutzen	7
Grundätzliches	7
Gründe für die Nutzung von R	7
Gründe	7
Übersicht - warum R	7
R lässt sich kombinieren	7
R für SPSS Nutzer	7
Die Popularität von R	7
R Nutzer rund um die Welt	7
Wo sind die aktivsten Nutzer?	7
Erwartungen und Anforderungen	7
Erwartungen und Anforderungen II	11
R herunterladen:	11
Links	11
Probleme mit Excel	12
Probleme mit Excel	12
Vergleich mit anderen Programmen	13
Dein Freund das GUI	13
Open Source Programm R	13
Graphisches User Interface	13
Verschiedene GUIs	13
Rstudio	14
Download der Unterlagen	14
Aufgabe - Vorbereitung	14
Grundlagen im Umgang mit der Sprache R	16
R ist eine Objekt-orientierte Sprache	16
Objektstruktur	16
Funktionen im base-Paket	16
Funktionen mit mehr Argumenten	17
Beispiel - Funktionen mit einem Argument	17
Funktionen mit einem Argument	17
Funktionen mit mehr Argumenten	17
Übersicht Befehle	18
Aufgabe - Zuweisungen und Funktionen	18
Datentypen und Indizieren	19
Verschiedene Datentypen	19
Verschiedene Datentypen	19
Indizieren eines Vektors:	19
data.frames	19
Indizieren	20
Matrizen und Arrays	20
Ein Array erzeugen	20

Indizieren eines Array	21
Listen	21
Indizieren einer Liste	21
Logische Operatoren	21
Operatoren um Subset für Datensatz zu bekommen	22
Datensätze einschränken	22
Weitere wichtige Optionen	22
Sequenzen	23
Weitere Sequenzen	23
Wiederholungen	23
Die Funktion paste	23
Wie bekommt man Hilfe?	24
Wie bekommt man Hilfe?	24
Nutzung Suchmaschinen	24
Stackoverflow	24
Ein Schummelzettel - Cheatsheet	25
Modularer Aufbau von R	25
Modularer Aufbau	25
Modularer Aufbau	25
Installation von Paketen mit RStudio	25
Vorhandene Pakete und Installation	25
Übersicht viele nützliche Pakete:	25
Pakete Regression	27
Big Data	27
Pakete von Github installieren	27
Wie bekomme ich einen Überblick	28
Aufgabe - Zusatzpakete	28
Datenimport	28
Datenimport	28
Dateiformate in R	28
Formate - base package	29
Der Arbeitsspeicher	29
Alternative - Arbeitsspeicher	29
Import von Excel-Daten	29
CSV Dateien einlesen	29
SPSS Dateien einlesen	31
stata Dateien einlesen	31
Das Paket rio	31
Datenmanagement ähnlich wie in SPSS oder Stata	31
Weitere Alternative Rcmdr	31
Aufgabe - Datenimport	31
Datenexport	32
Datenexport	32
R's Exportformate	32
Beispieldatensatz erzeugen	32
Überblick Daten Import/Export	32
Daten in Excel Format abspeichern	32
Daten in stata Format abspeichern	33
Auch zum Export eignet sich das rio Paket	33
Links Export	33

Exkurs: Datenquellen	34
Datenzugang	34
Datenquellen	34
Das R-Paket datasets	34
Datensatz zum US Zensus	34
Weltbank Daten	34
Nutzung von WDI Daten	35
Erste Grafik mit WDI Daten	35
OpenStreetMap	35
Download von OpenStreetMap Daten	35
TwittR	35
worldHires Daten	35
Historische Daten	35
GDELT Daten	36
Andere Datenquellen	36
Weitere Quellen	36
Datenanalyse	36
Streuungsmaße	36
Extremwerte	37
Fehlende Werte	37
Häufigkeiten und gruppierte Kennwerte	37
Tabellieren - weiteres Beispiel	38
Eine weitere Tabelle	38
Häufigkeitstabellen	38
Die Funktion <code>prop.table</code>	39
Die aggregate Funktion	39
Beispieldatensatz - apply Funktion	39
Die Funktion <code>apply</code>	40
Argumente der Funktion <code>apply</code>	40
Die Funktion <code>tapply</code>	40
Beispiel Funktion <code>tapply</code>	40
Links Datenanalyse	41
Aufgabe - Apply Funktion anwenden	41
Einfache Grafiken	41
Ein Plot sagt mehr als 1000 Worte	41
Plot ist nicht gleich Plot	41
CRAN Task Views	42
Task View zu Thema Graphiken	42
Datensatz	42
Histogramm - Die Funktion <code>hist()</code>	43
Graphik speichern	43
Befehl um Graphik zu speichern	43
Histogramme	45
Histogramm	45
Barplot	46
Barplots und barcharts	46
Mehr Farben:	46
Grüne Farbe	47
Rote Farbe	48
Transparent	49
Boxplot	50
Horizontaler Boxplot	50

Gruppierte Boxplots	51
Beispiel grupperter Boxplot	51
Alternativen zu Boxplot	52
Die Bibliothek <code>vioplot</code>	52
<code>vioplot</code> - Das Ergebnis	53
Alternativen zum Boxplot	53
Grafiken für bedingte, bi- und multivariate Verteilungen	54
Scatterplots	54
Aufgabe - einfache Grafiken	54
Zusammenhang	55
Edgar Anderson's Iris Daten	55
Zusammenhang zwischen stetigen Variablen	56
Zusammenhang zwischen mehreren Variablen	56
Zusammenhang zwischen mehreren Variablen	56
Verschiedene Korrelationskoeffizienten	57
Zusammenhang zwischen kategorialen Variablen	58
Levelplot	58
Visualisierung von Zusammenhängen zwischen kategorialen Variablen	58
Shading	59
Literatur zu Zusammenhangsmaßen	60
Das lattice Paket	60
Das lattice-Paket	60
Histogramm mit Lattice	60
Histogramm mit Lattice	61
Die Dichte mit Lattice zeichnen	62
Boxplot mit Lattice zeichnen	63
Boxplot mit Lattice zeichnen	64
Univariate Plots	65
Densityplot	66
Bivariate Plots	67
xyplot	68
Multivariate Plots	69
parallelplot	71
Lattice Befehle	72
Aufgabe - OECD Datensatz	72
Die lineare Regression	73
Die lineare Regression	73
Lineare Regression in R - Beispieldatensatz	73
Das lineare Regressionsmodell in R	73
Summary des Modells	74
R arbeitet mit Objekten	74
Residuenplot	75
Residuenplot	75
Linkliste - lineare Regression	76
Aufgabe - lineare Regression	76
Die logistische Regression	77
Agresti - Categorical Data Analysis (2002)	77
Faraway Bücher zu Regression in R	77
Binäre AVs mit <code>glm</code>	78
Beispieldaten für die logistische Regression	78

Logistische Regression mit R	78
Logistische Regression mit R	79
Weitere Beispieldaten	79
Generalisierte Regression mit R - weitere Funktionen	79
Linkliste - logistische Regression	79
Aufgabe - Datenanalyse	80
Faktoren in R	80
Was sind Faktoren in R	80
Beispiel Definition von Faktoren	80
Weitere Möglichkeit der Definition	81
Beispiel Monate	81
Beispiel: ordered factor	81
Rücktransformation	81
Tabellen mit Faktoren	82
Grafiken mit ggplot	82
Das Paket <code>ggplot2</code>	82
Basiseinführung <code>ggplot2</code>	82
Der <code>diamonds</code> Datensatz	82
Wie nutzt man <code>qplot</code>	83
Ein Balkendiagramm	83
Ein weiteres Balkendiagramm	84
Boxplot	85
Scatterplot	86
Farbe hinzufügen	87
Trendlinie hinzufügen	88
Graphik drehen	89
Wie nutzt man <code>ggplot</code>	90
Farben selber wählen	91
Eine Graphik mit den gewählten Farben	91
Speichern mit <code>ggsave</code>	92
Links	92
Einfache Karten mit R erstellen	93
Gliederung	93
Straßenkarten	93
Installieren des Paketes	93
Paket <code>ggmap</code> - Hallo Welt	94
Karte für eine Sehenswürdigkeit	94
Karte für einen ganzen Staat	95
Ein anderes <i>zoom level</i>	96
Hilfe bekommen wir mit dem Fragezeichen	97
Die Beispiele in der Hilfe	97
Ein anderes <i>zoom level</i>	98
Näher rankommen	98
Ganz nah dran	99
<code>ggmap</code> - <code>maptype satellite</code>	100
<code>ggmap</code> - <code>maptype satellite zoom 20</code>	101
<code>ggmap</code> - <code>maptype hybrid</code>	102
Terrain/physical maps	103
<code>ggmap</code> - <code>terrain map</code>	103
Abstrahierte Karten (http://www.designfaves.com)	104
<code>ggmap</code> - <code>maptype watercolor</code>	105

ggmap - source stamen	105
ggmap - maptype toner-lite	106
ggmap - maptype toner-hybrid	107
ggmap - maptype terrain-lines	108
Graphiken speichern	109
ggmap - ein Objekt erzeugen	109
Geokodierung	109
Latitude und Longitude	110
Koordinaten verschiedener Orte in Deutschland	110
Reverse Geokodierung	110
Die Distanz zwischen zwei Punkten	110
Eine andere Distanz bekommen	110
Geokodierung - verschiedene Punkte von Interesse	111
Punkte in der Karte	111
Punkte in der Karte	112
ggmap - verschiedene Farben	113
ggmap - größere Punkte	114
Eine Route von Google maps bekommen	115
Eine Karte mit dieser Information zeichnen	115
Cheatsheet	116
Resourcen und Literatur	116
Take Home Message	116
Regressionsdiagnostik mit R-Paket visreg	119
Regressionsdiagnostik mit Basis-R	119
Modellvorhersage machen	119
Regressionsdiagnostik mit Basis-R	119
Das visreg -Paket	120
Visualisierung	120
Und dann mit visreg visualisiert.	122
Visualisierung mit dem Paket visreg	122
Das visreg -Paket	123
Regression mit Faktoren	124
Effekte von Faktoren	124
Das Paket visreg - Interaktionen	125
Steuern der Graphikausgabe mittels layout	125
Das Paket visreg - Interaktionen overlay	126
Das Paket visreg - visreg2d	127
Das Paket visreg - surface	128
Weitere Themen im Ausblick	129
Mehr User Interface - Der R-commander	129
Interaktive Grafiken mit R	129
R-Paket rggobi	130
Interaktion mit plots	130
Tabellen für Publikationen	130
Tabellen mit dem R-Paket knitr	130
Nichtlineare Regression	130
Beispiel einer interaktiven Karte	131

Warum R nutzen

Grundätzliches

- Meistens sind die Kenntnisse und Fähigkeiten der Teilnehmer sehr heterogen - bitte sagen, wenn es zu schnell oder langsam geht
- Wenn Fragen sind - immer fragen
- R macht zusammen mehr Spaß - gerne den Nachbarn fragen

Gründe für die Nutzung von R

- Als Weg kreativ zu sein ...
- Graphiken, Graphiken, Graphiken
- In Kombination mit anderen Programmen nutzbar
- Zur Verbindung von Datenstrukturen
- Zum Automatisieren
- Um die Intelligenz anderer Leute zu nutzen ;-)
- ...

Gründe

- R ist frei verfügbar. Es kann umsonst runtergeladen werden.
- R ist eine Skriptsprache
- Gute Möglichkeiten für die Visualisierung (Link)
- R wird immer populärer
- Popularität von R

Übersicht - warum R

R lässt sich kombinieren...

R für SPSS Nutzer

```
install.packages("Rcmdr")
library("Rcmdr")
```

Bob Munich - R for SPSS and SAS Users

Die Popularität von R

R Nutzer rund um die Welt

Wo sind die aktivsten Nutzer?

Erwartungen und Anforderungen

Das kann diese Schulung vermitteln:

- Eine praxisnahe Einführung in die statistische Programmiersprache R

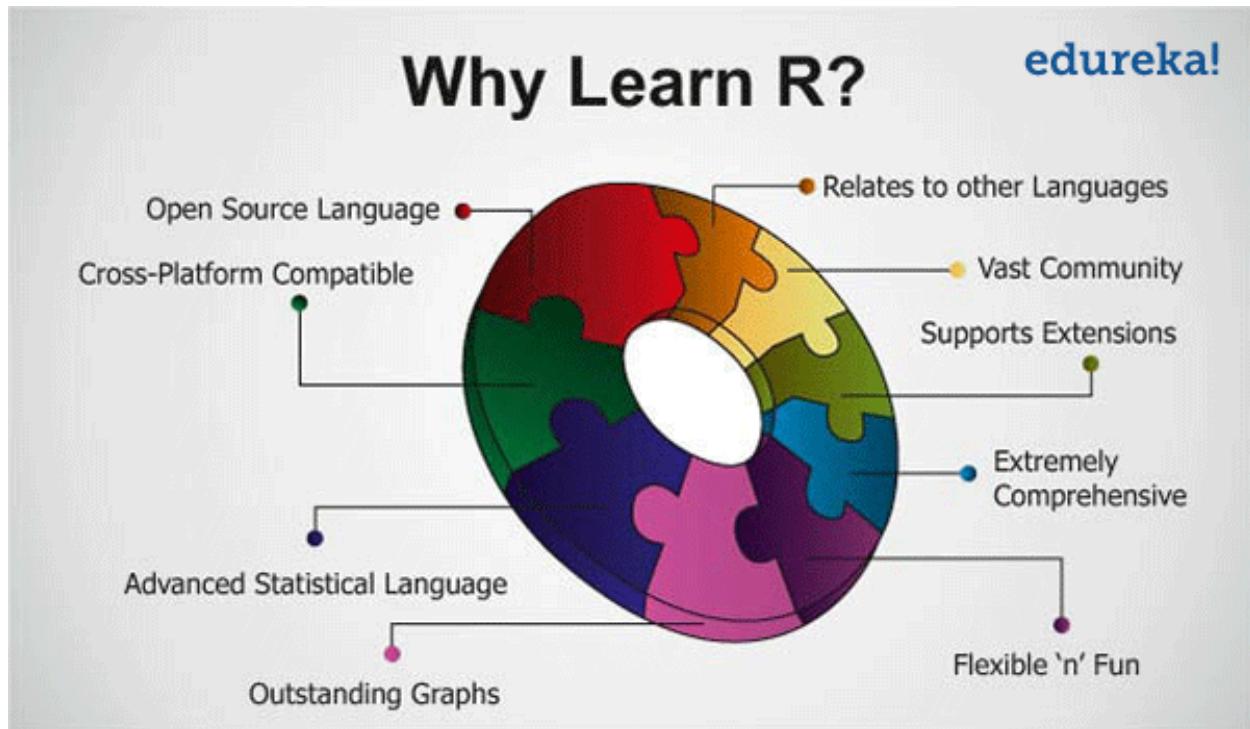


Figure 1:

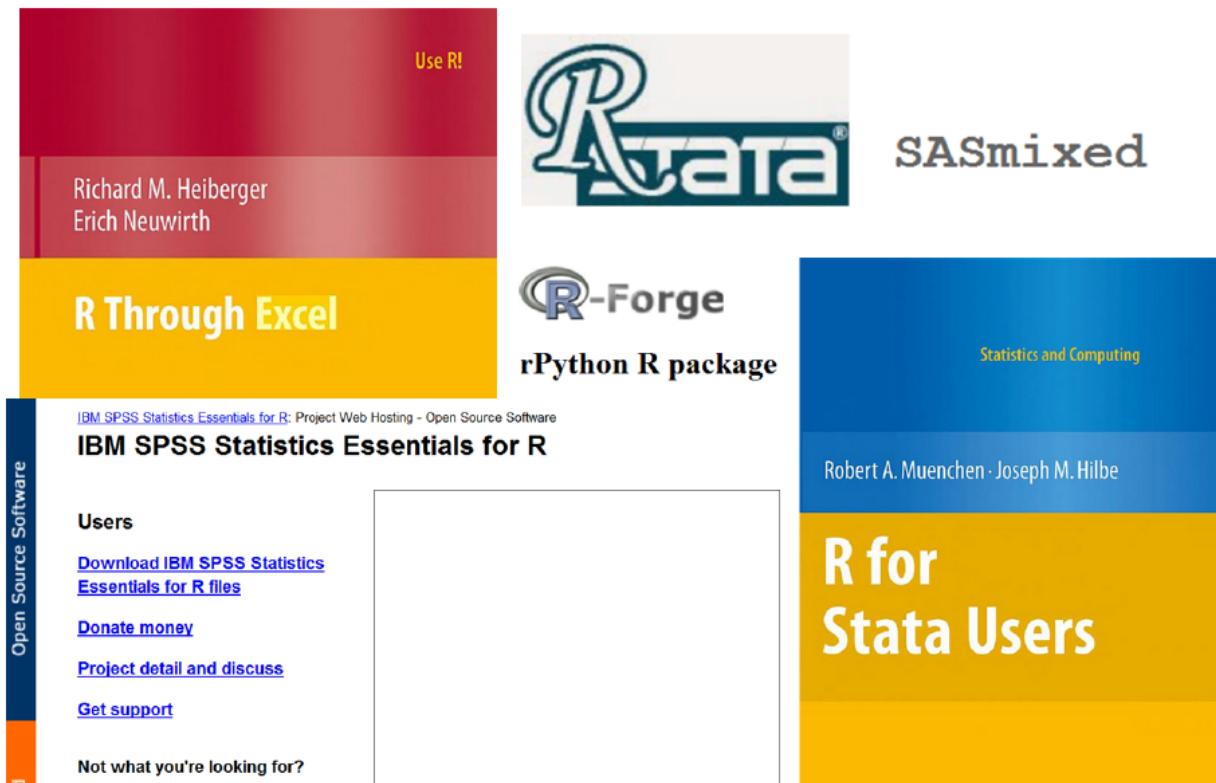


Figure 2:

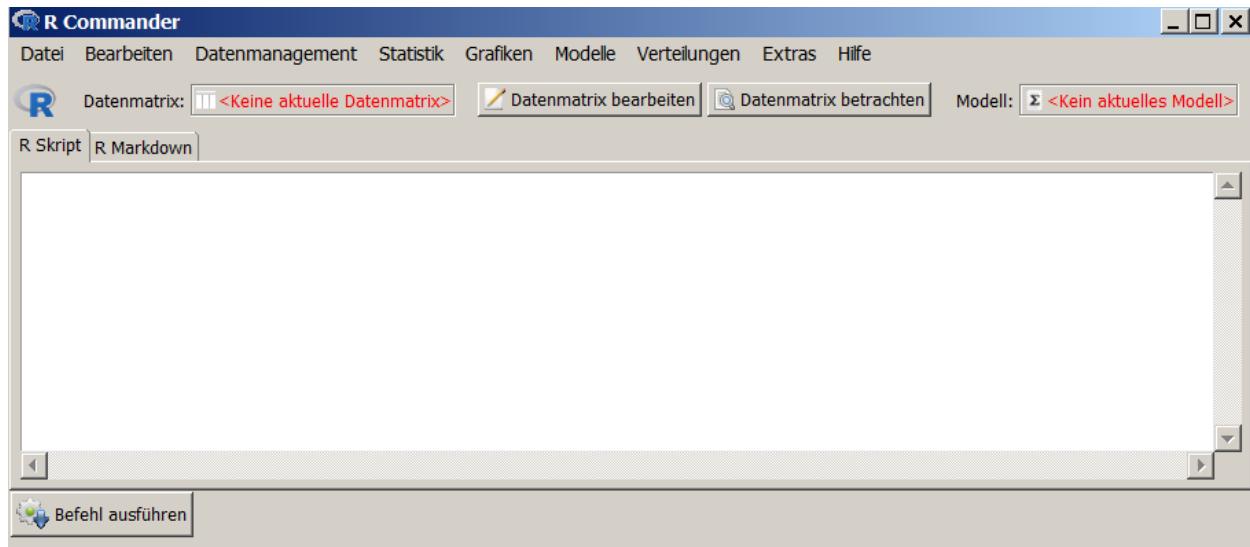


Figure 3:

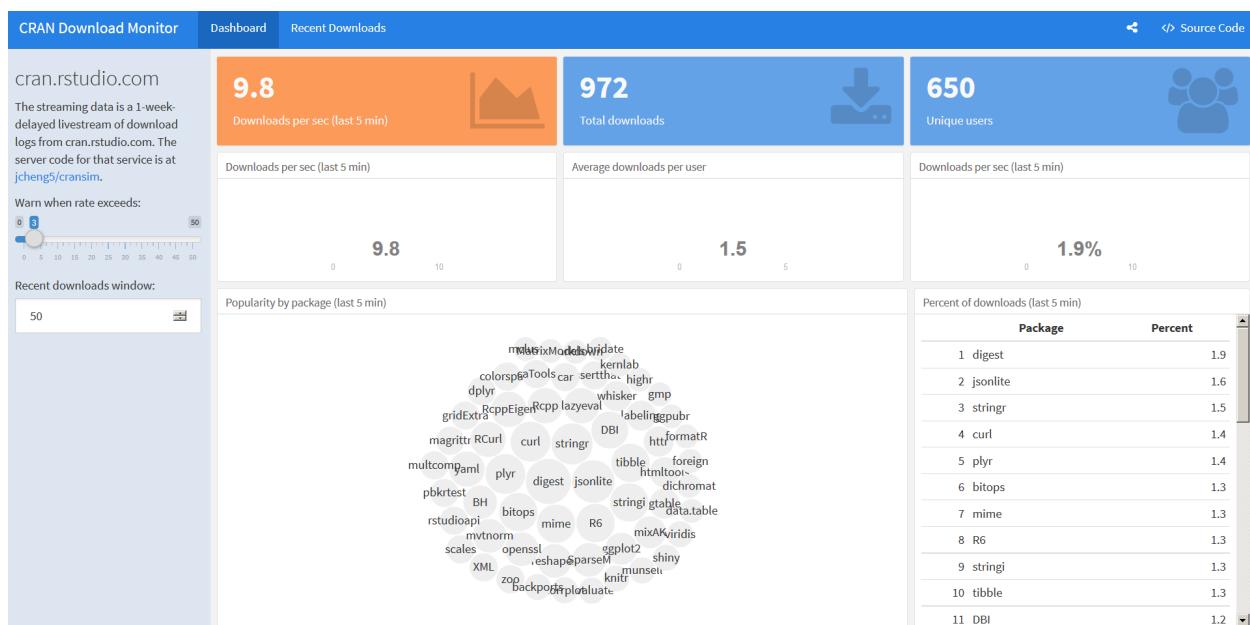


Figure 4:

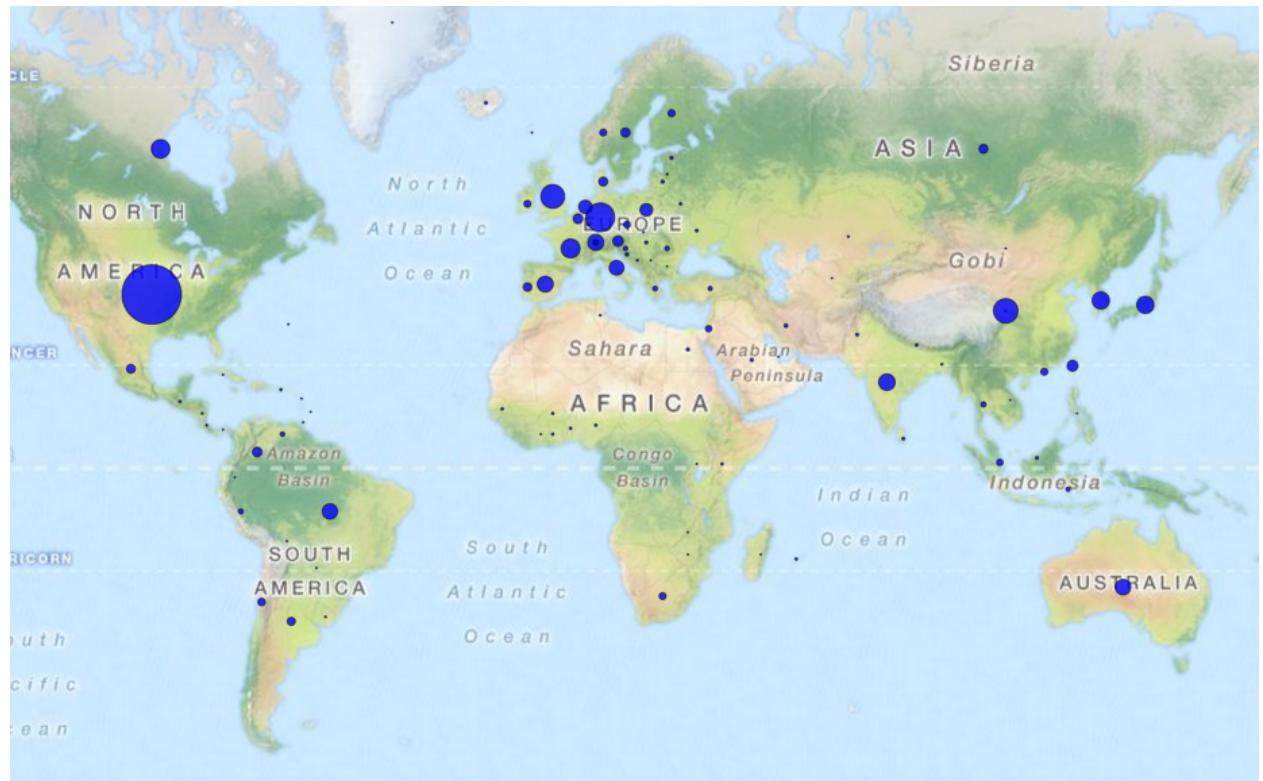


Figure 5: R Welt

R Activity Around the World

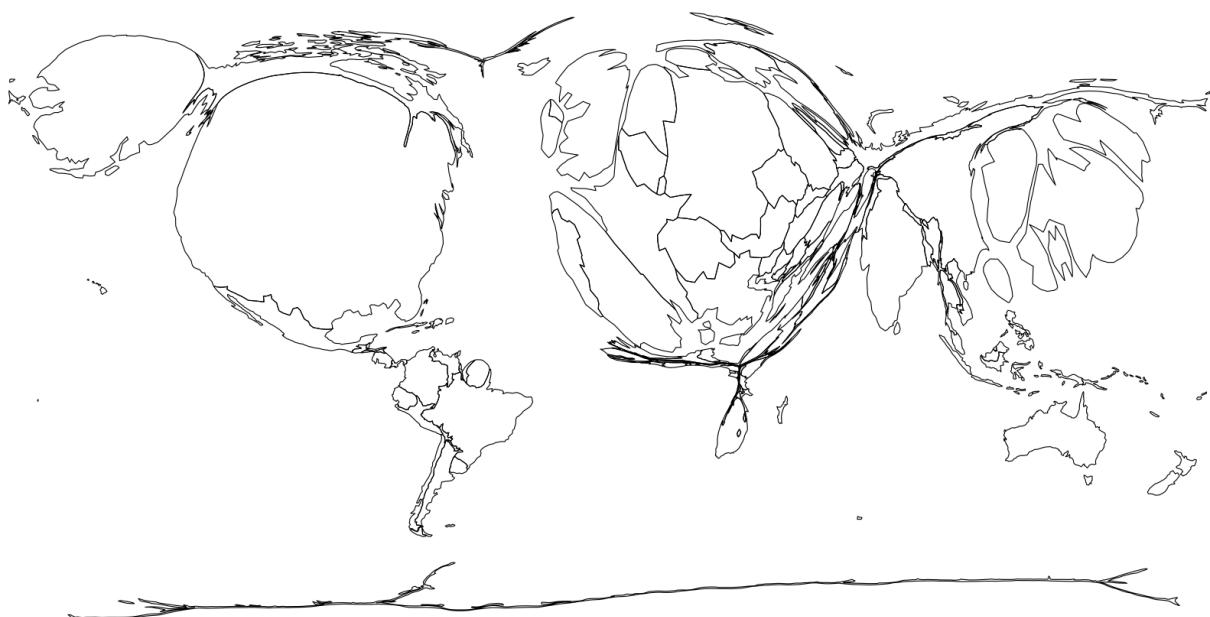


Figure 6: Aktivität Nutzer

- Erlernen einer Programmier-Strategie
- Guten Stil
- Die Vorzüge graphischer Datenanalyse

Erwartungen und Anforderungen II

Das kann sie nicht leisten:

- Eine Einführungsveranstaltung in die Statistik geben
- Grundlegende datenanalytische Konzepte vermitteln
- Verständnis zementieren
- Das Trainieren abnehmen

R herunterladen:

<http://www.r-project.org/>



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows** and **Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Friday 2017-04-21, You Stupid Darkness)
[R-3.4.0.tar.gz](#), read [what's new](#) in the latest version.

Figure 7:

Links

- Warum man R für Data Science lernen sollte
- R Technologie des Jahres
- Why R is Good for Business
- Warum R auf r-bloggers
- Intro R
- Intro R II

- Vergleich python und R

Probleme mit Excel

Weil andere Programme große Fehler haben:

- Excel bug
- Datum in Excel



Figure 8:

Probleme mit Excel

BioMed Central

BMC Bioinformatics

HOME ABOUT ARTICLES SUBMISSION GUIDELINES

CORRESPONDENCE | OPEN ACCESS

Mistaken Identifiers: Gene name errors can be introduced inadvertently when using Excel in bioinformatics

Barry R Zeeberg[†], Joseph Riss[†], David W Kane, Kimberly J Bussey, Edward Uchio, W Marston Linehan, J Carl Barrett and John N Weinstein

[†]Contributed equally

BMC Bioinformatics 2004 5:80 | DOI: 10.1186/1471-2105-5-80 | © Zeeberg et al; licensee BioMed Central Ltd. 2004
Received: 05 March 2004 | Accepted: 23 June 2004 | Published: 23 June 2004

Abstract

Figure 9:

Weitere Teile der Artikelserie zu Stärken und Schwächen gängiger Statistik-Software:

- Checkliste für die Anschaffung von Statistik-Software
- **Statistik-Software: R, SAS, SPSS und STATA im Vergleich**
- Die mächtige Open Source-Lösung: R

Statistik-Software: R, SAS, SPSS und STATA im Vergleich

Figure 10:

Vergleich mit anderen Programmen

Dein Freund das GUI

Open Source Programm R

- R ist eine freie, nicht-kommerzielle Implementierung der Programmiersprache S (von AT&T Bell Laboratories entwickelt)
- Freie Beteiligung - modularer Aufbau (immer mehr Erweiterungspakete)
- Der Download ist auf dieser Seite möglich:

<https://cran.r-project.org/>

Graphisches User Interface

Aber die meisten Menschen nutzen einen Editor oder ein graphical user interface (GUI).

Aus den folgenden Gründen:

- Syntax highlighting
- Auto-Vervollständigung
- Bessere Übersicht über Graphiken, Bibliotheken

Verschiedene GUIs

- Gedit mit R-spezifischen Add-ons für Linux
- Emacs
- TinnR
- Ich nutze Rstudio!

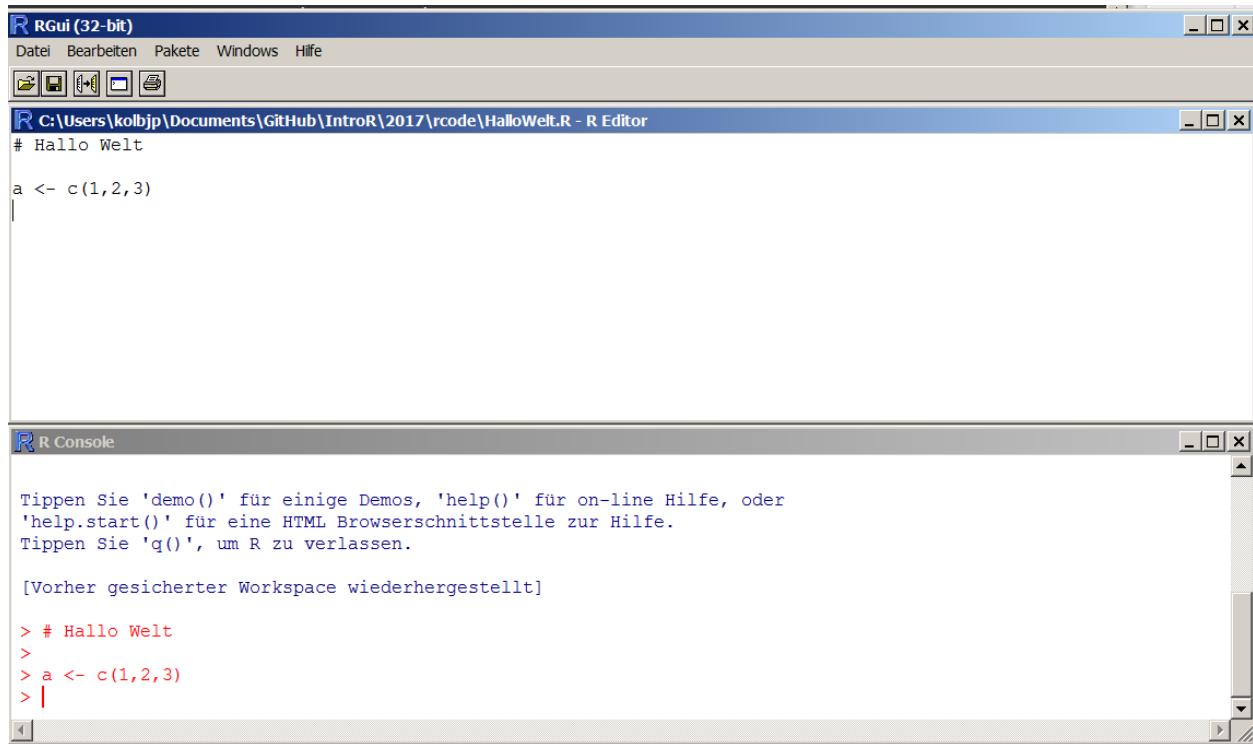


Figure 11:

Rstudio

- Sechs Gründe Rstudio zu nutzen.
- Wie man Rstudio nutzen kann.
- Das Rstudio einrichten

Download der Unterlagen

Auf github sind alle Unterlagen für diesen Kurs zu finden.

Wie nutzt man github?

Aufgabe - Vorbereitung

- Prüfen Sie, ob eine Version von R auf Rechner installiert ist.
- Falls dies nicht der Fall ist, laden Sie R runter und installieren Sie R.
- Prüfen Sie, ob Rstudio installiert ist.
- Falls nicht - Installieren sie Rstudio.
- Laden Sie die R-Skripte von meinem GitHub-Account
- Erstellen Sie ein erstes Script und finden Sie das Datum mit dem Befehl `date()` und die R-version mit `sessionInfo()` heraus.

```
date()
```

```
## [1] "Tue May 02 12:48:06 2017"
```

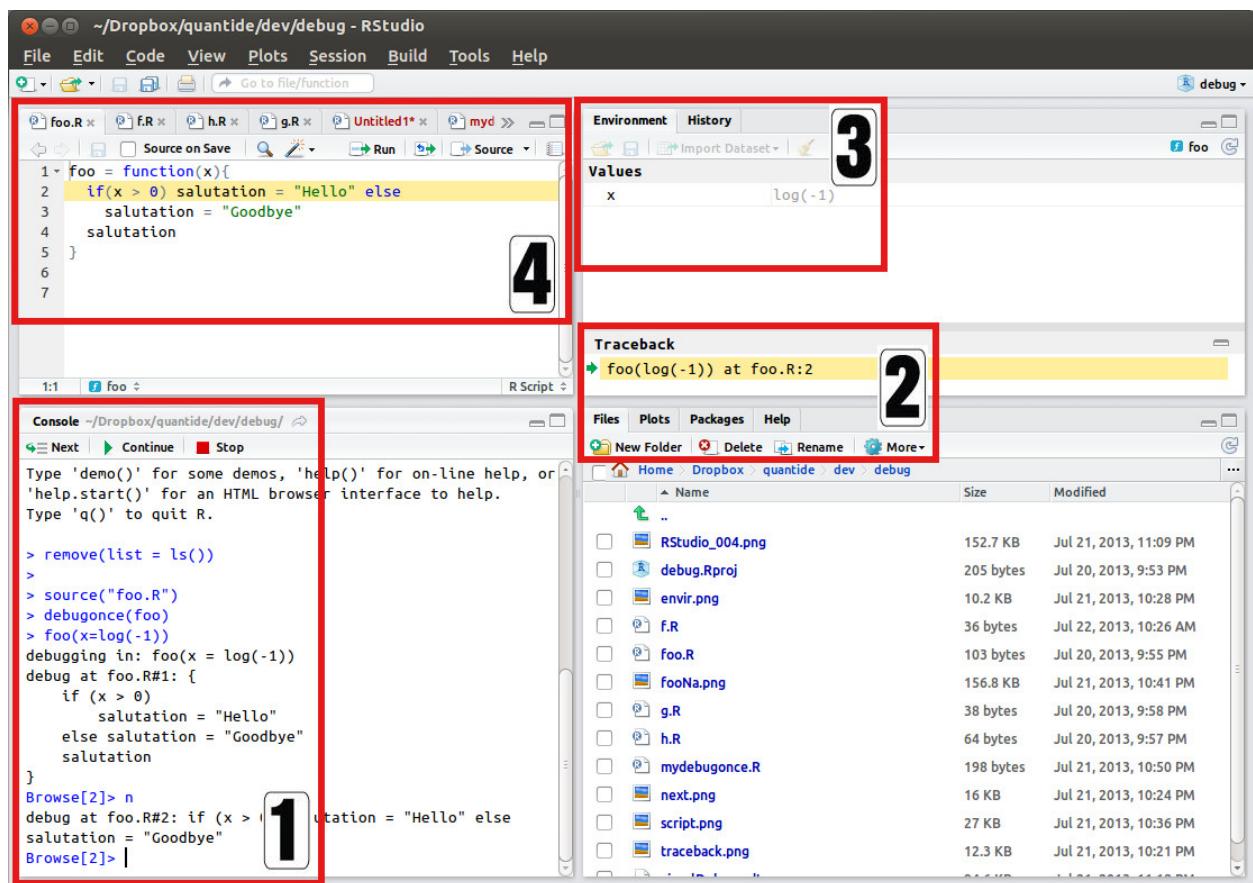


Figure 12: rstudio

```

sessionInfo()

## R version 3.3.2 (2016-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats      graphics   grDevices utils      datasets   methods    base
##
## loaded via a namespace (and not attached):
## [1] backports_1.0.4 magrittr_1.5     rprojroot_1.1   tools_3.3.2
## [5] htmltools_0.3.5 yaml_2.1.14    Rcpp_0.12.9    stringi_1.1.2
## [9] rmarkdown_1.3   knitr_1.15.1   stringr_1.2.0   digest_0.6.11
## [13] evaluate_0.10

```

Grundlagen im Umgang mit der Sprache R

R ist eine Objekt-orientierte Sprache

Vektoren und Zuweisungen

- R ist eine Objekt-orientierte Sprache
- <- ist der Zuweisungsoperator

```
b <- c(1,2) # erzeugt ein Objekt mit den Zahlen 1 und 2
```

- Eine Funktion kann auf dieses Objekt angewendet werden:

```
mean(b) # berechnet den Mittelwert
```

```
## [1] 1.5
```

Mit den folgenden Funktionen können wir etwas über die Eigenschaften des Objekts lernen:

```
length(b) # b hat die Länge 2
```

```
## [1] 2
```

Objektstruktur

```
str(b) # b ist ein numerischer Vektor
```

```
## num [1:2] 1 2
```

Funktionen im base-Paket

Funktion	Bedeutung	Beispiel
length()	Länge	length(b)

Funktion	Bedeutung	Beispiel
max()	Maximum	max(b)
min()	Minimum	min(b)
sd()	Standardabweichung	sd(b)
var()	Varianz	var(b)
mean()	Mittelwert	mean(b)
median()	Median	median(b)

Diese Funktionen brauchen nur ein Argument.

Funktionen mit mehr Argumenten

Andere Funktionen brauchen mehr:

Argument	Bedeutung	Beispiel
quantile()	90 % Quantile	quantile(b,.9)
sample()	Stichprobe ziehen	sample(b,1)

Beispiel - Funktionen mit einem Argument

```
max(b)
## [1] 2
min(b)
## [1] 1
sd(b)
## [1] 0.7071068
var(b)
## [1] 0.5
```

Funktionen mit einem Argument

```
mean(b)
## [1] 1.5
median(b)
## [1] 1.5
```

Funktionen mit mehr Argumenten

```
quantile(b,.9)
```

```
## 90%
## 1.9
sample(b,1)
## [1] 2
```

Übersicht Befehle

<http://cran.r-project.org/doc/manuals/R-intro.html>

An Introduction to R

Table of Contents

Preface

1 Introduction and preliminaries

- 1.1 The R environment
- 1.2 Related software and documentation
- 1.3 R and statistics
- 1.4 R and the window system
- 1.5 Using R interactively
- 1.6 An introductory session
- 1.7 Getting help with functions and features
- 1.8 R commands, case sensitivity, etc.
- 1.9 Recall and correction of previous commands
- 1.10 Executing commands from or diverting output to a file
- 1.11 Data permanency and removing objects

Figure 13:

Aufgabe - Zuweisungen und Funktionen

Erzeugen Sie einen Vektor b mit den Zahlen von 1 bis 5 und berechnen Sie...

1. den Mittelwert
2. die Varianz
3. die Standardabweichung
4. die quadratische Wurzel aus dem Mittelwert

Datentypen und Indizieren

Verschiedene Datentypen

Datentyp	Beschreibung	Beispiel
numeric	ganze und reelle Zahlen	5, 3.462
logical	logische Werte	FALSE, TRUE
character	Buchstaben und Zeichenfolgen	“Hallo”

Quelle: R. Münnich und M. Knobelgespieß (2007): Einführung in das statistische Programm Paket R

Verschiedene Datentypen

```
b <- c(1,2) # numeric
log <- c(T,F) # logical
char <- c("A","b") # character
fac <- as.factor(c(1,2)) # factor
```

Mit `str()` bekommt man den Objekttyp.

Indizieren eines Vektors:

```
A1 <- c(1,2,3,4)
A1

## [1] 1 2 3 4
A1[1]

## [1] 1
A1[4]

## [1] 4
A1[1:3]

## [1] 1 2 3
A1[-4]

## [1] 1 2 3
```

data.frames

Beispieldaten generieren:

```
AGE <- c(20,35,48,12)
SEX <- c("m","w","w","m")
```

Diese beiden Vektoren zu einem data.frame verbinden:

```
Daten <- data.frame(Alter=AGE,Geschlecht=SEX)
```

Anzahl der Zeilen/Spalten herausfinden

```
nrow(Daten) # Zeilen
```

```
## [1] 4
```

```
ncol(Daten) # Spalten
```

```
## [1] 2
```

Indizieren

Indizieren eines dataframes:

```
AA <- 4:1
```

```
A2 <- cbind(A1,AA)
```

```
A2[1,1]
```

```
## A1
```

```
## 1
```

```
A2[2,]
```

```
## A1 AA
```

```
## 2 3
```

```
A2[,1]
```

```
## [1] 1 2 3 4
```

```
A2[,1:2]
```

```
##      A1 AA
```

```
## [1,] 1 4
```

```
## [2,] 2 3
```

```
## [3,] 3 2
```

```
## [4,] 4 1
```

Matrizen und Arrays

- In Matrizen und Arrays stehen meist nur numerische Werte.
- Dadurch wird beispielsweise Matrix Multiplikation möglich.
- Anders als beim data.frame sind mehr als zwei Dimensionen möglich.

```
A <- matrix(seq(1,100), nrow = 4)
```

```
dim(A)
```

```
## [1] 4 25
```

Ein Array erzeugen

```
A3 <- array(1:8,c(2,2,2))
```

```
A3
```

```

## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8

```

Indizieren eines Array

```
A3[, , 2]
```

```

##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8

```

Listen

- Eine Liste in R entspricht einem geschachtelten Array in anderen Programmiersprachen
- Listen können alles enthalten
- Listen können geschachtelt sein
- Listen sollte man sehr bedacht verwenden

Indizieren einer Liste

```
A4 <- list(A1, 1)
A4
```

```

## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 1
A4[[2]]
```

```
## [1] 1
```

Logische Operatoren

```
# Ist 1 größer als 2?
1>2
```

```
## [1] FALSE
```

```
1<2
```

```
## [1] TRUE
```

```
1==2
```

```
## [1] FALSE
```

Operatoren um Subset für Datensatz zu bekommen

Diese Operatoren eignen sich gut um Datensätze einzuschränken

Daten

```
## Alter Geschlecht  
## 1 20 m  
## 2 35 w  
## 3 48 w  
## 4 12 m  
Daten[AGE>20,]
```

```
## Alter Geschlecht  
## 2 35 w  
## 3 48 w
```

Datensätze einschränken

```
Daten[SEX=="w",]
```

```
## Alter Geschlecht  
## 2 35 w  
## 3 48 w  
# gleiches Ergebnis:  
Daten[SEX!="m",]
```

```
## Alter Geschlecht  
## 2 35 w  
## 3 48 w
```

Weitere wichtige Optionen

```
# Ergebnis in ein Objekt speichern  
subDat <- Daten[AGE>20,]
```

```
# mehrere Bedingungen können mit  
# & verknüpft werden:  
Daten[AGE>18 & SEX=="w",]
```

```
## Alter Geschlecht  
## 2 35 w  
## 3 48 w
```

Sequenzen

```
# Sequenz von 1 bis 10
1:10

## [1] 1 2 3 4 5 6 7 8 9 10
Daten[1:3,]

## Alter Geschlecht
## 1      20      m
## 2      35      w
## 3      48      w
```

Weitere Sequenzen

```
seq(-2,8,by=1.5)

## [1] -2.0 -0.5  1.0  2.5  4.0  5.5  7.0
a <- seq(3,12,length=12)

b <- seq(to=5,length=12,by=0.2)

d <- 1:10
d <- seq(1,10,1)
d <- seq(length=10,from=1,by=1)
```

Wiederholungen

```
# wiederhole 1 10 mal
rep(1,10)

## [1] 1 1 1 1 1 1 1 1 1 1
rep("A",10)

## [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Die Funktion paste

```
?paste

paste(1:4)

## [1] "1" "2" "3" "4"
paste("A", 1:6, sep = "")

## [1] "A1" "A2" "A3" "A4" "A5" "A6"
```

Wie bekommt man Hilfe?

Wie bekommt man Hilfe?

- Um generell Hilfe zu bekommen:

```
help.start()
```

- Online Dokumentation für die meisten Funktionen:

```
help(name)
```

- Nutze ? um Hilfe zu bekommen.

```
?mean
```

- example(lm) gibt ein Beispiel für die lineare Regression

```
example(lm)
```

Nutzung Suchmaschinen

- Ich nutze meistens google
- Tippe:

R-project + Was ich schon immer wissen wollte

- Das funktioniert natürlich mit jeder Suchmaschine!

Stackoverflow

- Für Fragen zum Programmieren
- Ist nicht auf R fokussiert
- Sehr detaillierte Diskussionen

The screenshot shows the Stack Overflow homepage with the 'R' tag selected. At the top, there's a navigation bar with links for Questions, Jobs, Documentation (BETA), Tags, and Users. A search bar contains the query '[r]'. On the left, a sidebar lists 'Tagged Questions' with filters for info, newest, featured (selected), frequent, votes, active, and unanswered. Below this, a text box contains a brief description of R and links to learn more, top users, synonyms, and jobs. The main content area shows a question titled 'How to make a great R reproducible example?' with 1776 votes, 22 answers, and 147K views. The question text asks for guidance on making reproducible examples. To the right, there's a sidebar for the 'R Language' tag, showing 22,187 frequent questions and a link to ask a new question. Another sidebar lists related tags: ggplot2, dataframe, and plot.

Figure 14:

Ein Schummelzettel - Cheatsheet

<https://www.rstudio.com/resources/cheatsheets/>



Figure 15:

Modularer Aufbau von R

Modularer Aufbau

Modularer Aufbau

- Viele Funktionen sind im Basis-R enthalten
- Viele spezifische Funktionen sind in zusätzlichen Bibliotheken integriert
- R kann modular erweitert werden durch sog. packages bzw. libraries
- Auf CRAN werden die wichtigsten packages gehostet (im Moment 10430)
- Weitergehende Pakete finden sich z.B. bei bioconductor

```
install.packages("lme4")
library(lme4)
```

Installation von Paketen mit RStudio

Vorhandene Pakete und Installation

Übersicht viele nützliche Pakete:

- Luhmann - Tabelle mit vielen nützlichen Paketen

Die wichtigsten Pakete zur Visualisierung mit R:

- ggplot
- lattice
- Visualisierung kategorialer Daten
- Interaktive Visualisierungen
- plotrix

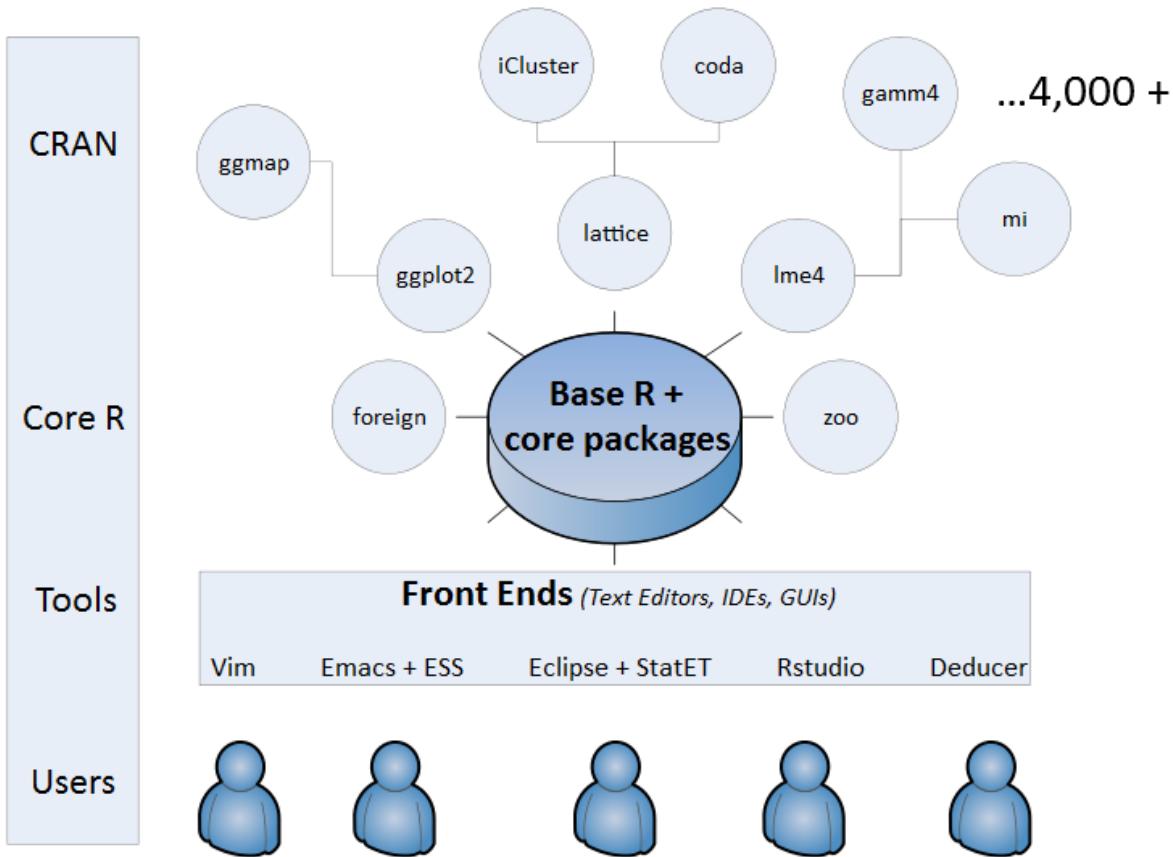


Figure 16:

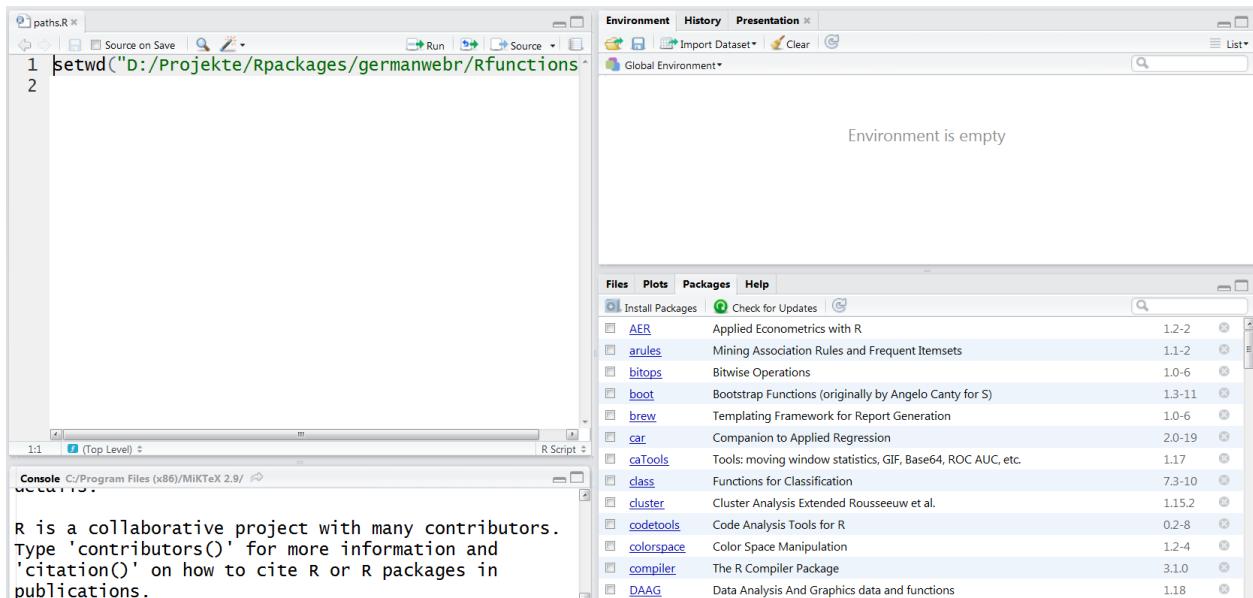


Figure 17:

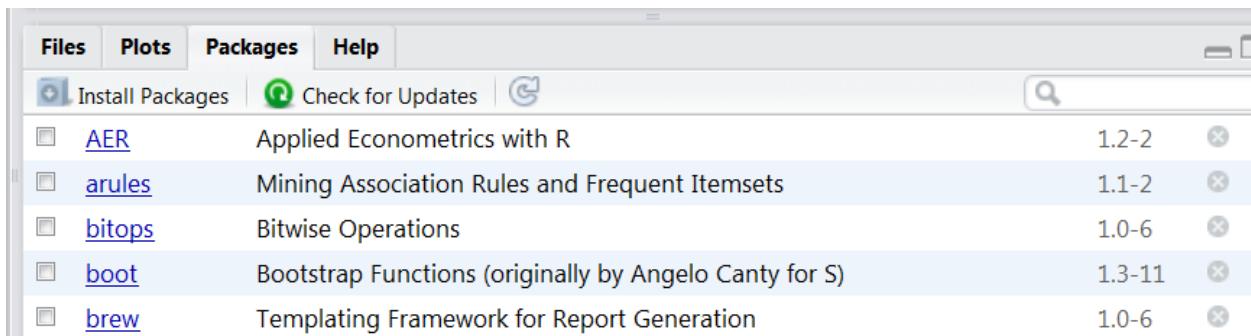


Figure 18:

- Farbpaletten in R

Pakete Regression

- R-Paket MASS
- Autoregressive Modelle (Zeitreihen)
- Robuste Regressionen
- Nichtparametrische Regression
- Lasso Verfahren

Big Data

- Task View - High Performance Computing

Weitere interessante Pakete

Paket für den Import/Export - foreign

- Pakete für Survey Sampling
- Paket - Latex und R (xtable) (xtable Galerie)
- Paket zur Erzeugung von Dummies
- Multivariate Normalverteilung
- Paket für Karten

Pakete von Github installieren

```
install.packages("devtools")
library(devtools)

install_github("hadley/ggplot2")
```

Wie bekomme ich einen Überblick

- Explore Packages Currently on CRAN
- Pakete die in letzter Zeit von CRAN heruntergeladen wurden

Aufgabe - Zusatzpakete

Gehen Sie auf <cran.r-project.org> und suchen Sie in dem Bereich, wo die Pakete vorgestellt werden, nach Paketen,...

- die für die deskriptive Datenanalyse geeignet sind.
- um Regressionen zu berechnen
- um fremde Datensätze einzulesen (z.B. SPSS-Daten)
- um mit großen Datenmengen umzugehen

Datenimport

Datenimport



Figure 19:

Dateiformate in R

- Von R werden quelloffene, nicht-proprietary Formate bevorzugt

- Es können aber auch Formate von anderen Statistik Software Paketen eingelesen werden
- R-user speichern Objekte gerne in sog. Workspaces ab
- Auch hier jedoch gilt: (fast) alles andere ist möglich

Formate - base package

R unterstützt von Haus aus schon einige wichtige Formate:

- CSV (Comma Separated Values): `read.csv()`
- FWF (Fixed With Format): `read.fwf()`
- Tab-getrennte Werte: `read.delim()`

Der Arbeitsspeicher

So findet man heraus, in welchem Verzeichnis man sich gerade befindet

```
getwd()
```

So kann man das Arbeitsverzeichnis ändern:

Man erzeugt ein Objekt in dem man den Pfad abspeichert:

```
main.path <- "C:/" # Beispiel für Windows
main.path <- "/users/Name/" # Beispiel für Mac
main.path <- "/home/user/" # Beispiel für Linux
```

Und ändert dann den Pfad mit `setwd()`

```
setwd(main.path)
```

Bei Windows ist es wichtig Slashes anstelle von Backslashes zu verwenden.

Alternative - Arbeitsspeicher

Import von Excel-Daten

- `library(foreign)` ist für den Import von fremden Datenformaten nötig
- Wenn Excel-Daten vorliegen - als .csv abspeichern
- Dann kann `read.csv()` genutzt werden um die Daten einzulesen.
- Bei Deutschen Daten kann es sein, dass man `read.csv2()` wegen der Komma-Separierung braucht.

```
library(foreign)
?read.csv
?read.csv2
```

CSV Dateien einlesen

Zunächst muss das Arbeitsverzeichnis gesetzt werden, in dem sich die Daten befinden:

```
Dat <- read.csv("schuldaten_export.csv")
```

Wenn es sich um Deutsche Daten handelt:

```
Dat <- read.csv2("schuldaten_export.csv")
```

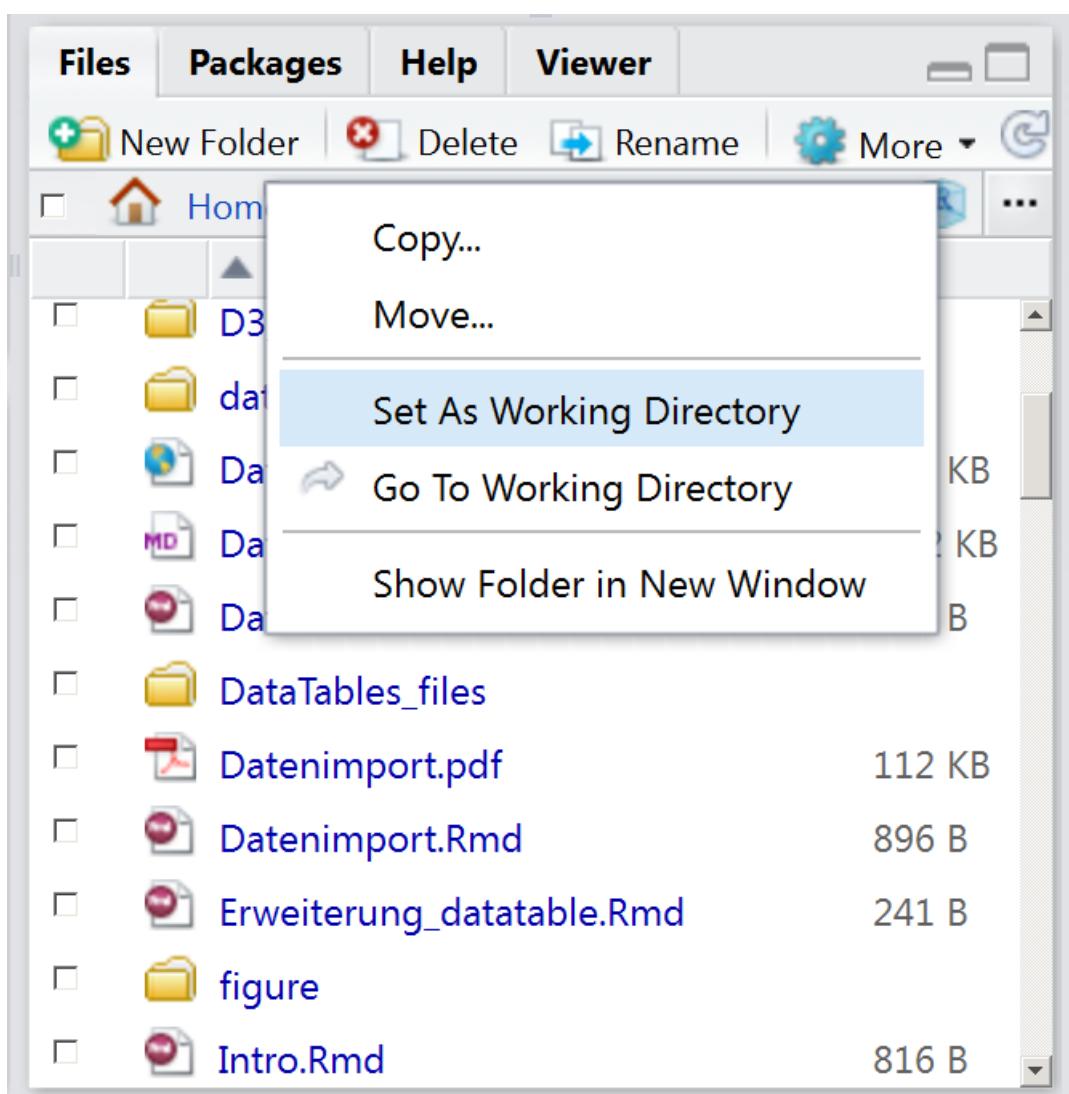


Figure 20:

SPSS Dateien einlesen

Dateien können auch direkt aus dem Internet geladen werden:

```
link<- "http://www.statistik.at/web_de/static/  
mz_2013_sds_-_datensatz_080469.sav"  
  
?read.spss  
Dat <- read.spss(link,to.data.frame=T)
```

stata Dateien einlesen

```
MZ02 <- read.dta("MZ02.dta")
```

- Einführung in Import mit R (is.R)

Das Paket rio

```
install.packages("rio")  
  
library("rio")  
x <- import("mtcars.csv")  
y <- import("mtcars.rds")  
z <- import("mtcars.dta")
```

- rio: A Swiss-Army Knife for Data I/O

Datenmanagement ähnlich wie in SPSS oder Stata

```
install.packages("Rz")  
library(Rz)
```

Weitere Alternative Rcmdr

```
install.packages("Rcmdr")
```

- Funktioniert auch mit Rstudio

```
library(Rcmdr)
```

Aufgabe - Datenimport

- Gehen Sie auf meine Github Seite und laden Sie den OECD Datensatz herunter
- Laden Sie den Datensatz mit einer geeigneten Funktion in Ihre Console.
- Finden Sie heraus, wieviele Beobachtungen und Variablen der Datensatz umfasst.

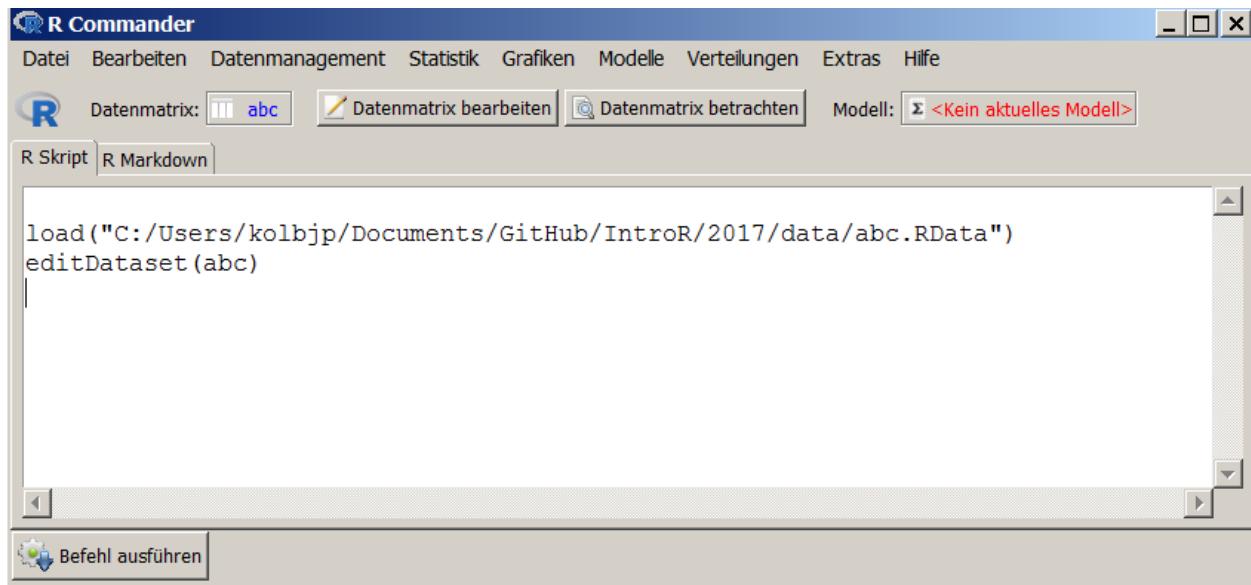


Figure 21:

Datenexport

Datenexport

R's Exportformate

- In R werden offene Dateiformate bevorzugt
- Als Äquivalenz zu den `read.X()` Funktionen stehen viele `write.X()` Funktionen zur Verfügung
- Das eigene Format von R sind sog. Workspaces (`.RData`)

Beispieldatensatz erzeugen

```
A <- c(1,2,3,4)
B <- c("A", "B", "C", "D")

mydata <- data.frame(A,B)
```

Überblick Daten Import/Export

```
save(mydata, file="mydata.RData")
```

Daten in Excel Format abspeichern

```
write.csv(mydata,file="mydata.csv")

library(xlsx)
write.xlsx(mydata,file="mydata.xlsx")
```



Figure 22:

Daten in stata Format abspeichern

```
library(foreign)
write.dta(mydata, file="mydata.dta")
```

Auch zum Export eignet sich das rio Paket

```
library("rio")

export(mtcars, "mtcars.csv")
export(mtcars, "mtcars.rds")
export(mtcars, "mtcars.dta")
```

Links Export

- Quick R für das Exportieren von Daten:
- Hilfe zum Export auf dem cran Server
- Zurück zur Gliederung.

Exkurs: Datenquellen

Datenzugang

- Public-Use-File (PUF) Datei zur öffentlichen Nutzung - meist stark anonymisierte Daten (Beispiele: FDZ, Statistik Portal, Meine Region)
- Scientific-Use-File (SUF) - Datei zur wissenschaftlichen Nutzung - anonymisierte Daten, die zu wissenschaftlichen Zwecken und zur Sekundäranalyse genutzt werden können.
- On-Site-Nutzung - Arbeitsplätze für Gastwissenschaftler - Kontrollierte Datenfernverarbeitung

Datenquellen

- Auf dem Portal datahub.io sind sehr viele Beispieldatensätze in verschiedenen Formaten abrufbar.
- Weitere Portale: OpenGov, okfn, enigma, Amazon Web Services (AWS)
- Umweltdaten (National climatic data center)
- FAO Datenbank

```
library("FAOSTAT")
```

- Public Use File für Soziales in den USA Social security administration
- National health and nutrition examination survey

```
library(survey)
data(nhanes)
```

Das R-Paket datasets

```
library(datasets)
```

Beispiel Erdbeben Datensatz:

```
head(quakes)
```

Datensatz zum US Zensus

```
library(UScensus2010)
```

Weltbank Daten

WDI - World Development Indicators (World Bank) - Einführung in das Paket

```
library(WDI)
```

```
WDIsearch('gdp')[1:10,]
```

Nutzung von WDI Daten

```
dat <- WDI(indicator='NY.GDP.PCAP.KD', country=c('MX','CA','US'), start=1960, end=2012)
head(dat)
```

Erste Grafik mit WDI Daten

OpenStreetMap

OpenStreetMap (OSM) ist ein kollaboratives Projekt um eine editierbare Weltkarte zu erzeugen.
Wikipedia - OpenStreetMap

Download von OpenStreetMap Daten

```
library(osmar)
api <- osmsource_api()
library(ggmap)

cityC <- geocode("Berlin",source="google")
bb <- center_bbox(cityC$lon,cityC$lat,1000, 1000)
uaBerlin <- get_osm(bb, source = api)
```

- Ausschnitte von OpenStreetMap für einzelne Städte (metro extracts)
- Liste möglicher Datenquellen für räumliche Analysen (weltweit, Deutschland)
- SALB - Administrative Grenzen
- Kartendaten (openaprs)

TwittR

```
library(twitteR)
library(streamR)
```

<http://www.r-bloggers.com/mapping-the-world-with-tweets-including-a-gif-without-cats-and-a-shiny-app/>

worldHires Daten

```
library(mapdata)
data(worldHiresMapEnv)
map('worldHires', col=1:10)
```

Historische Daten

- Historischer Geocoder
- Paket HistData

```
library(HistData)
data(Arbuthnot)
```

GDELT Daten

- GDELT
- Nutzung von GDELT Daten (Beispiel 1, Beispiel 2)

```
library(GDELTtools)
test.filter <- list(ActionGeo_ADM1Code=c("NI", "US"), ActionGeo_CountryCode="US")
test.results <- GetGDELT(start.date="1979-01-01", end.date="1979-12-31",
                        filter=test.filter)
```

Andere Datenquellen

- Die US Flughäfen und Fluglinien
- Mehr Daten hier

```
link1 <- "http://openflights.svn.sourceforge.net/viewvc/openflights/
          openflights/data/airports.dat"
airport <- read.csv(link1, header = F)

link2 <- "http://openflights.svn.sourceforge.net/viewvc/openflights/
          openflights/data/routes.dat"
route <- read.csv(link2, header = F)
```

- Hafen Daten (Natural earth data)
- Minimalistische Karten
- Census results - Germany
- Census results - Britain and boundaries
- Data on airports and an example on the usage in R
- ADFC/opengeodb

```
link <- "http://www.fa-technik.adfc.de/code/opengeodb/DE9.tab"
info <- read.csv(link, sep="\t", header=F)
```

Weitere Quellen

- ICEDS European Data Server
- Mobilfunkdaten, CO2 Emmissionen
- Daten für New York (Daten, Beispiel)

Datenanalyse

Streuungsmaße

Im base Paket sind die wichtigsten Streuungsmaße enthalten:

- Varianz: `var()`
- Standardabweichung: `sd()`
- Minimum und Maximum: `min()` und `max()`
- Range: `range()`

```
ab <- rnorm(100); var(ab)
```

```
## [1] 0.9148144
```

```
sd(ab); range(ab)
```

```
## [1] 0.9564593
```

```
## [1] -2.059865 2.391989
```

Extremwerte

```
min(ab)
```

```
## [1] -2.059865
```

```
max(ab)
```

```
## [1] 2.391989
```

Fehlende Werte

- Sind NAs vorhanden muss dies der Funktion mitgeteilt werden

```
ab[10] <- NA
```

```
var(ab)
```

```
## [1] NA
```

Bei fehlenden Werten muss ein weiteres Argument mitgegeben werden:

```
var(ab,na.rm=T)
```

```
## [1] 0.917908
```

Häufigkeiten und gruppierte Kennwerte

- Eine Auszählung der Häufigkeiten der Merkmale einer Variable liefert `table()`
- Mit `table()` sind auch Kreuztabellierungen möglich indem zwei Variablen durch Komma getrennt werden: `table(x,y)` liefert Häufigkeiten von `y` für gegebene Ausprägungen von `x`

```
x <- sample(1:10,100,replace=T)
```

```
table(x)
```

```
## x
```

```
##  1  2  3  4  5  6  7  8  9 10
```

```
##  8  9 12  9  8 14  6 14  8 12
```

Tabellieren - weiteres Beispiel

```
musician <- sample(c("yes","no"),100,replace=T)

?table

table(x)

## x
##  1  2  3  4  5  6  7  8  9 10
##  8  9 12  9  8 14  6 14  8 12

table(x,musician)

##      musician
## x    no yes
## 1     5   3
## 2     5   4
## 3     6   6
## 4     6   3
## 5     6   2
## 6     7   7
## 7     2   4
## 8     4  10
## 9     4   4
## 10    9   3
```

Eine weitere Tabelle

```
data(esoph)
table(esoph$agegp)

##
## 25-34 35-44 45-54 55-64 65-74    75+
##    15     15     16     16     15     11
```

Häufigkeitstabellen

- `prop.table()` liefert die relativen Häufigkeiten
- Wird die Funktion außerhalb einer `table()` Funktion geschrieben erhält man die relativen Häufigkeiten bezogen auf alle Zellen

Die Funktion `prop.table()`

```
table(esoph$agegp,esoph$alcgp)
```

```
##
##          0-39g/day 40-79 80-119 120+
## 25-34            4     4     3     4
## 35-44            4     4     4     3
## 45-54            4     4     4     4
## 55-64            4     4     4     4
## 65-74            4     3     4     4
## 75+              3     4     2     2
```

Die Funktion `prop.table`

```
?prop.table  
  
prop.table(table(esoph$agegp,  
esoph$alcgp),1)  
  
##  
##      0-39g/day    40-79     80-119      120+  
## 25-34 0.2666667 0.2666667 0.2000000 0.2666667  
## 35-44 0.2666667 0.2666667 0.2666667 0.2000000  
## 45-54 0.2500000 0.2500000 0.2500000 0.2500000  
## 55-64 0.2500000 0.2500000 0.2500000 0.2500000  
## 65-74 0.2666667 0.2000000 0.2666667 0.2666667  
## 75+   0.2727273 0.3636364 0.1818182 0.1818182
```

Die aggregate Funktion

- Mit der `aggregate()` Funktion können Kennwerte für Untergruppen erstellt werden
- `aggregate(x,by,FUN)` müssen mindestens drei Argumente übergeben werden:

```
aggregate(state.x77,by=list(state.region),mean)  
  
##      Group.1 Population Income Illiteracy Life Exp Murder HS Grad  
## 1    Northeast 5495.111 4570.222 1.000000 71.26444 4.722222 53.96667  
## 2       South 4208.125 4011.938 1.737500 69.70625 10.581250 44.34375  
## 3 North Central 4803.000 4611.083 0.700000 71.76667 5.275000 54.51667  
## 4        West 2915.308 4702.615 1.023077 71.23462 7.215385 62.00000  
##      Frost Area  
## 1 132.7778 18141.00  
## 2 64.6250 54605.12  
## 3 138.8333 62652.00  
## 4 102.1538 134463.00
```

x: ein oder mehrere Beobachtungsvektor(en) für den der Kennwert berechnet werden soll

by: eine oder mehrere bedingende Variable(n)

FUN: die Funktion welche den Kennwert berechnet (z.B. `mean` oder `sd`)

- Die Ausgabe kann mit Hilfe von `xtabs()` in eine schöne zweidimensionale Tabelle überführt werden

Beispieldatensatz - apply Funktion

```
ApplyDat <- cbind(1:4,runif(4),rnorm(4))  
  
apply(ApplyDat,1,mean)  
  
## [1] 0.3017396 0.8851795 1.0814802 1.7478401  
apply(ApplyDat,2,mean)  
  
## [1] 2.5000000 0.54304978 -0.03087023
```

Die Funktion apply

```
apply(ApplyDat,1,var)

## [1] 0.3995513 1.0250492 3.1003167 3.8099497

apply(ApplyDat,1,sd)

## [1] 0.6321007 1.0124471 1.7607716 1.9519092

apply(ApplyDat,1,range)

##          [,1]      [,2]      [,3]      [,4]
## [1,] -0.2314438 0.02292402 -0.4606845 0.5457233
## [2,]  1.0000000 2.00000000 3.0000000 4.0000000

apply(ApplyDat,1,length)

## [1] 3 3 3 3
```

Argumente der Funktion apply

- Für `margin=1` die Funktion `mean` auf die Reihen angewendet,
- Für `margin=2` die Funktion `mean` auf die Spalten angewendet,
- Anstatt `mean` können auch andere Funktionen wie `var`, `sd` oder `length` verwendet werden.

Die Funktion tapply

```
ApplyDat <- data.frame(Income=rnorm(5,1400,200),
                        Sex=sample(c(1,2),5,replace=T))
```

- Auch andere Funktionen können eingesetzt werden. . . - Auch selbst programmierte Funktionen
- Im Beispiel wird die einfachste eigene Funktion angewendet.

```
ApplyDat
```

```
##   Income Sex
## 1 1585.486  2
## 2 1376.690  2
## 3 1451.833  1
## 4 1346.355  2
## 5 1564.398  1
```

Beispiel Funktion tapply

```
tapply(ApplyDat$Income,ApplyDat$Sex,mean)
```

```
##      1       2
## 1508.116 1436.177
```

```
tapply(ApplyDat$Income,
       ApplyDat$Sex,function(x)x)
```

```

## $`1`
## [1] 1451.833 1564.398
##
## $`2`
## [1] 1585.486 1376.690 1346.355

```

Links Datenanalyse

- Die Benutzung von `apply`, `tapply`, etc. (Artikel bei R-bloggers)
- Quick-R zu deskriptiver Statistik
- Quick-R zur Funktion `aggregate`

Aufgabe - Apply Funktion anwenden

- Erstellen Sie eine Matrix A mit 4 Zeilen und 25 Spalten, die die Werte 1 bis 100 enthält. Analog dazu erstellen Sie eine Matrix B mit 25 Zeilen und 4 Spalten, die die Werte 1 bis 100 enthält.
- Berechnen Sie mittels dem `apply()`-Befehl den Mittelwert und die Varianz für jede Zeile von A bzw. B.
- Berechnen Sie mittels dem `apply()`-Befehl den Mittelwert und die Varianz für jede Spalte von A bzw. B.
- Standardisieren ist eine häufige Transformation von Daten; dafür wird der Mittelwert von der entsprechenden Zeile oder Spalte abgezogen und durch die entsprechende Standardabweichung geteilt. Somit besitzen die Daten einen Mittelwert von 0 und eine Standardabweichung von 1. Standardisieren Sie die Spalten der Matrix A .

Zurück zur Gliederung.

Einfache Grafiken

Ein Plot sagt mehr als 1000 Worte

- Grafisch gestützte Datenanalyse ist toll
- Gute Plots können zu einem besseren Verständnis beitragen
- Einen Plot zu generieren geht schnell
- Einen guten Plot zu machen kann sehr lange dauern
- Mit R Plots zu generieren macht Spaß
- Mit R erstellte Plots haben hohe Qualität
- Fast jeder Plottyp wird von R unterstützt
- R kennt eine große Menge an Exportformaten für Grafiken

Plot ist nicht gleich Plot

- Bereits das base Package bringt eine große Menge von Plot Funktionen mit
- Das lattice Packet erweitert dessen Funktionalität
- Eine weit über diese Einführung hinausgehende Übersicht findet sich in Murrell, P (2006): R Graphics.

CRAN Task Views

- Zu einigen Themen sind alle Möglichkeiten in R zusammengestellt. (Übersicht der Task Views)
- Zur Zeit gibt es 35 Task Views
- Alle Pakete eines Task Views können mit folgendem Befehl installiert werden:

```
install.packages("ctv")
library("ctv")
install.views("Bayesian")
```

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance

Figure 23:

Task View zu Thema Graphiken

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer: Nicholas Lewin-Koh
Contact: nikko at hailmail.net
Version: 2015-01-07
URL: <https://CRAN.R-project.org/view=Graphics>

R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including coplots, mosaic plots, biplots, and the list goes on. There are devices such as postscript, png, jpeg and pdf for outputting graphics as well as device drivers for all platforms running R. [lattice](#) and grid are supplied with R's recommended packages and are included in every binary distribution. [lattice](#) is an R implementation of William Cleveland's trellis graphics, while grid defines a much more flexible graphics environment than the base R graphics.

Figure 24:

Datensatz

```
library(mlmRev)
data(Chem97)

• [lea] Local Education Authority - a factor
• [school] School identifier - a factor
• [student] Student identifier - a factor
• [score] Point score on A-level Chemistry in 1997
• [gender] Student's gender
• [age] Age in month, centred at 222 months or 18.5 years
```

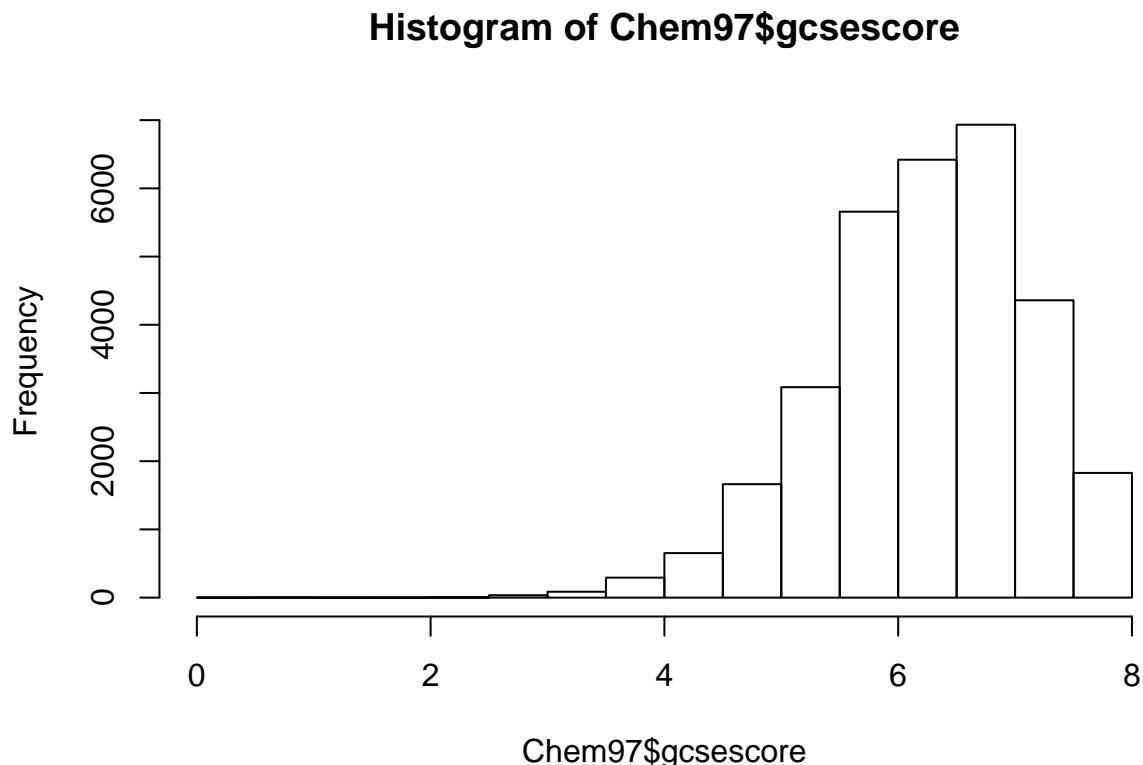
- [gcsescore] Average GCSE score of individual.
- [gcsecnt] Average GCSE score of individual, centered at mean.

Histogramm - Die Funktion hist()

Wir erstellen ein Histogramm der Variable gcsescore:

```
?hist
```

```
hist(Chem97$gcsescore)
```



Graphik speichern

- Mit dem button Export in Rstudio kann man die Grafik speichern.

Befehl um Graphik zu speichern

- Alternativ auch bspw. mit den Befehlen png, pdf oder jpeg

```
png("Histogramm.png")
hist(Chem97$gcsescore)
dev.off()
```

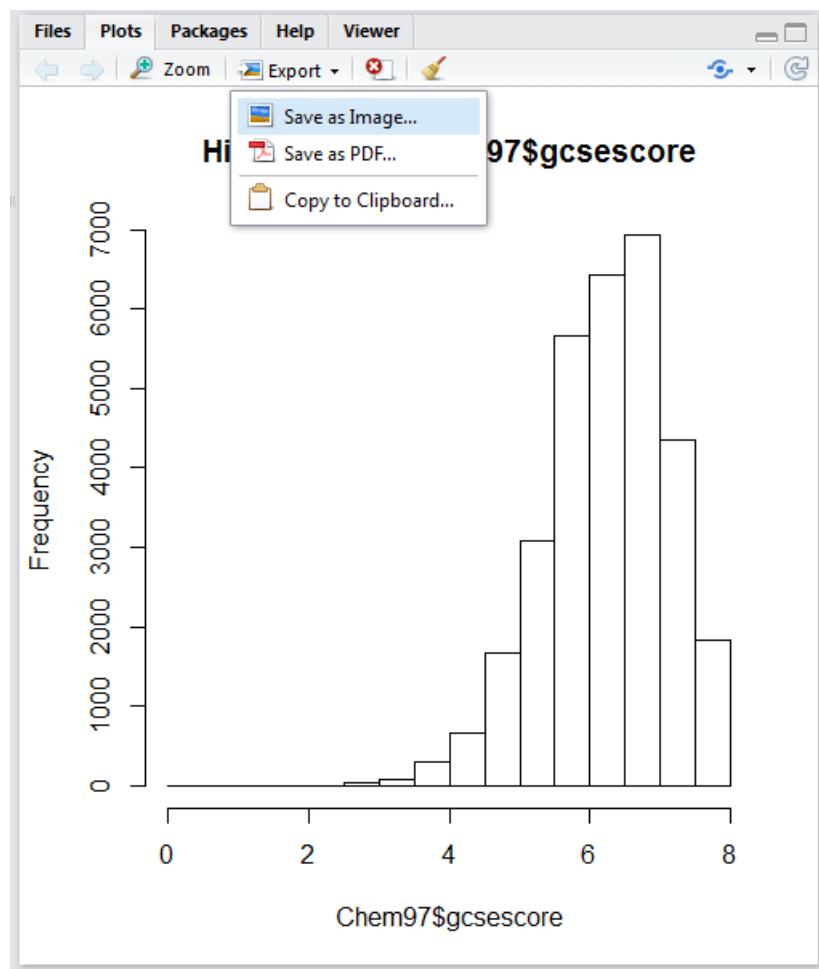


Figure 25:

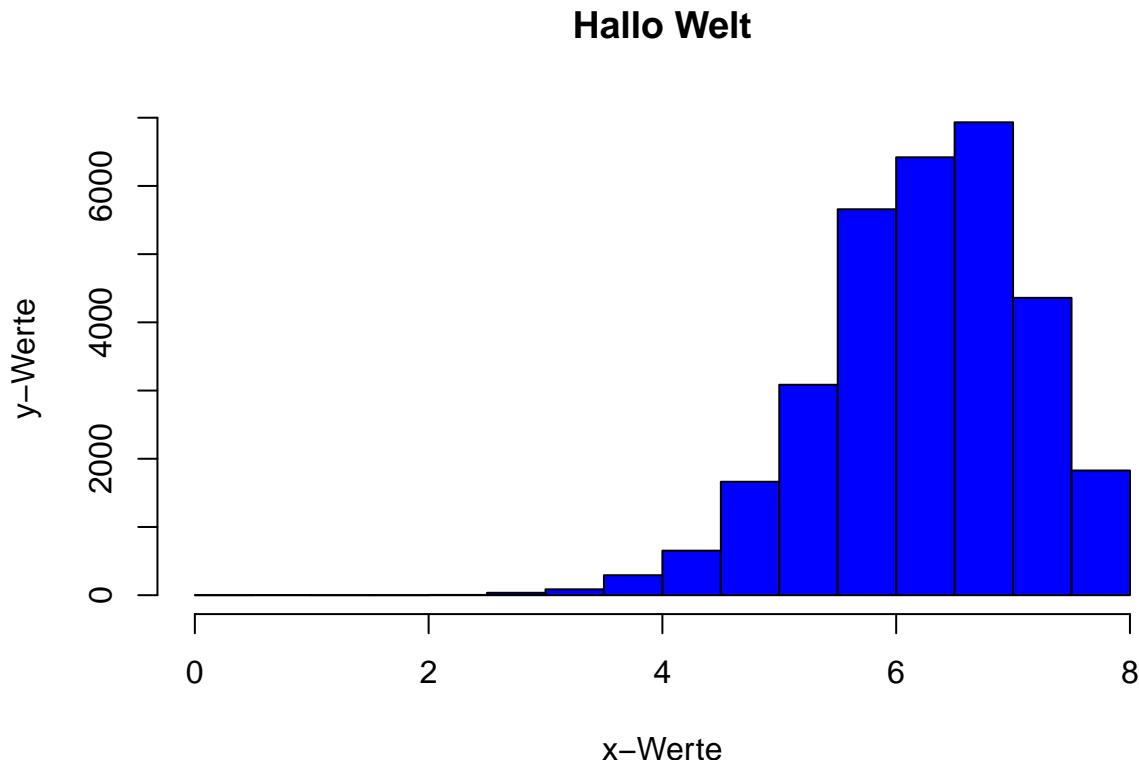
Histogramme

- Die Funktion `hist()` plottet ein Histogramm der Daten
- Der Funktion muss mindestens ein Beobachtungsvektor übergeben werden
- `hist()` hat noch sehr viel mehr Argumente, die alle (sinnvolle) default values haben

Argument	Bedeutung	Beispiel
main	Überschrift	main="Hallo Welt"
xlab	x-Achsenbeschriftung	xlab="x-Werte"
ylab	y-Achsenbeschriftung	ylab="y-Werte"
col	Farbe	col="blue"

Histogramm

```
hist(Chem97$gcsescore,col="blue",
     main="Hallo Welt",ylab="y-Werte", xlab="x-Werte")
```



Weitere Argumente:

```
?plot  
# oder  
?par
```

Barplot

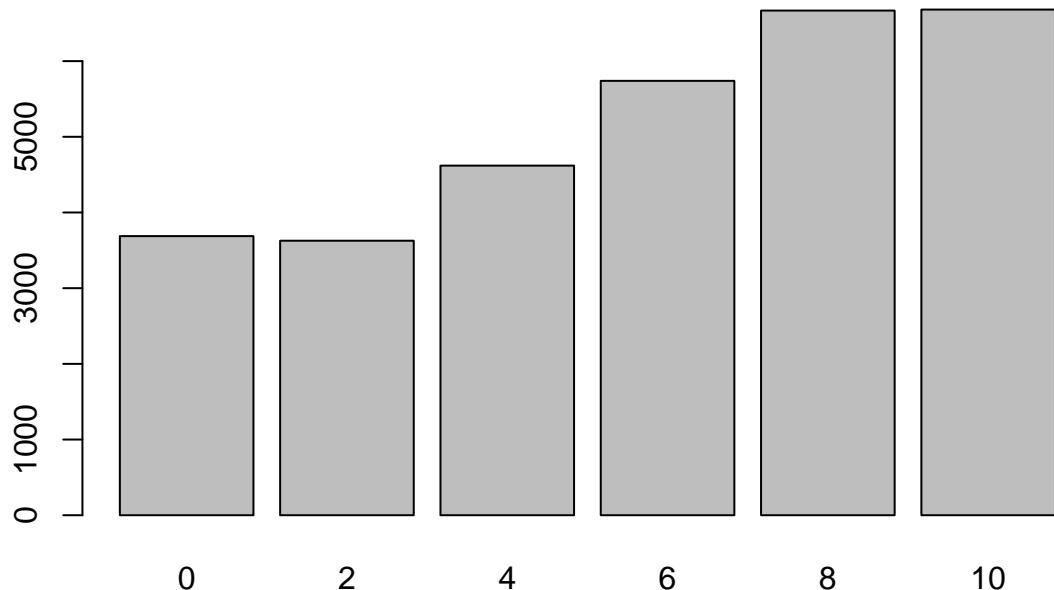
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

```
tabScore <- table(Chem97$score)
```

```
barplot(tabScore)
```

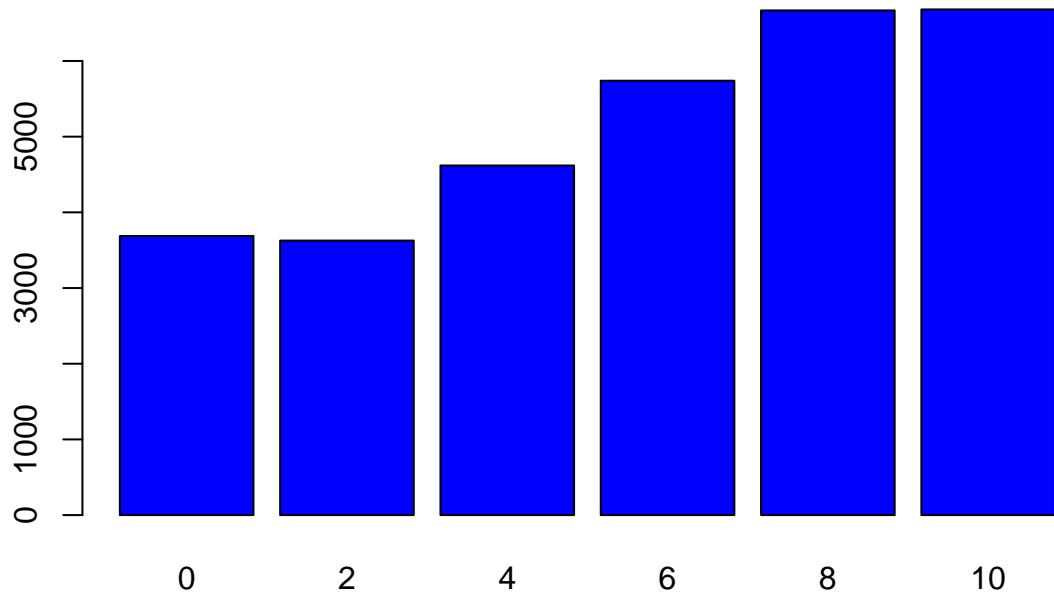
Barplots und barcharts

```
barplot(tabScore)
```



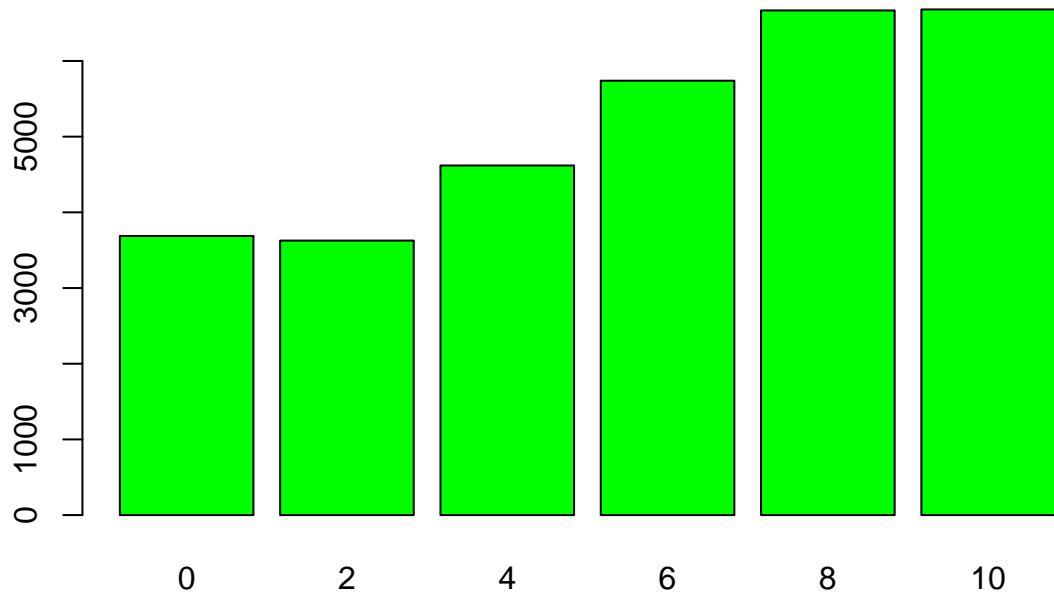
Mehr Farben:

```
barplot(tabScore, col=rgb(0,0,1))
```



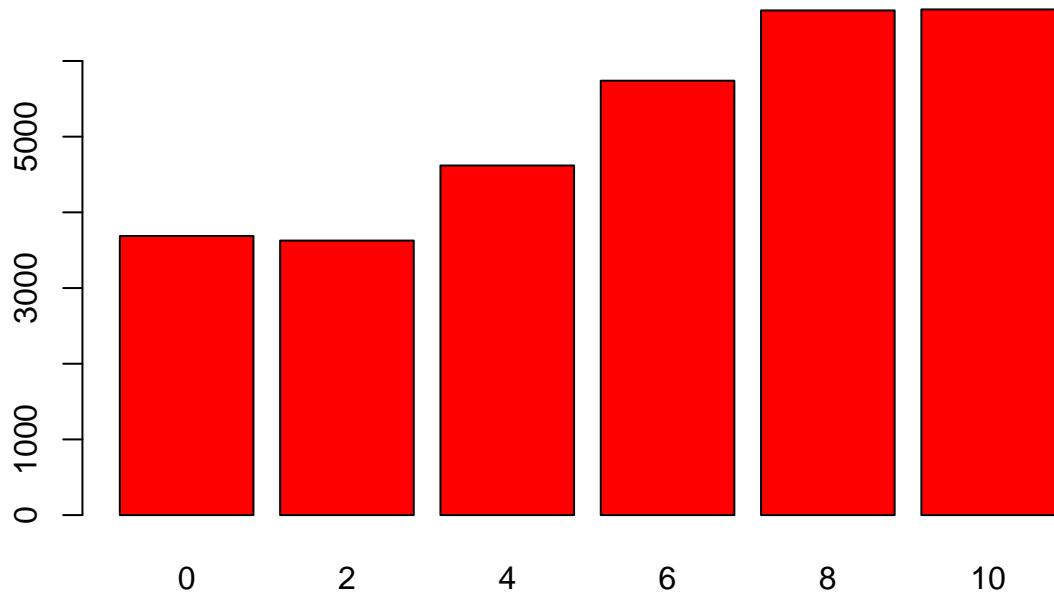
Grüne Farbe

```
barplot(tabScore,col=rgb(0,1,0))
```



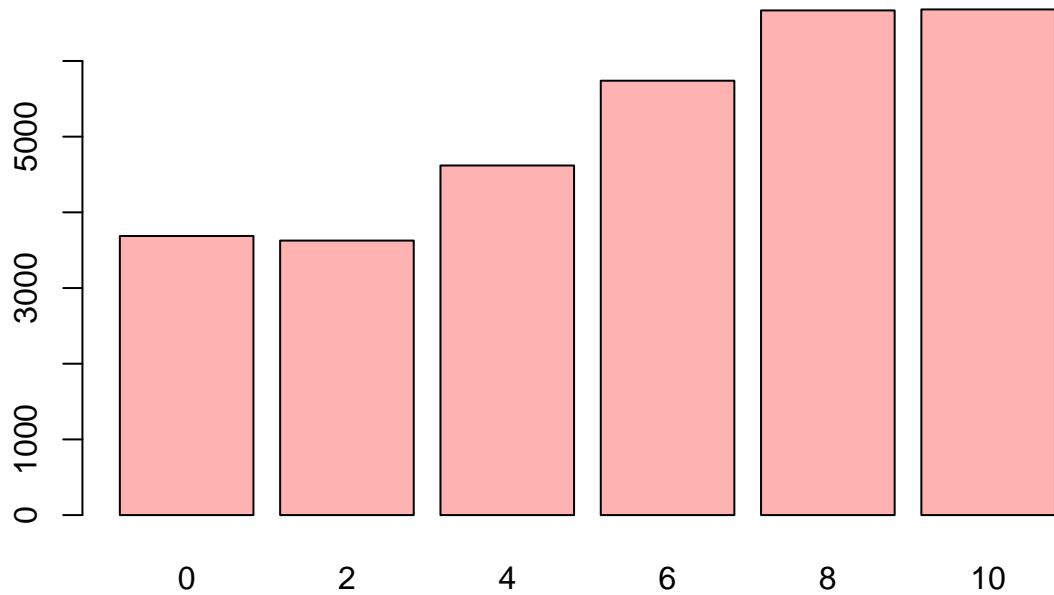
Rote Farbe

```
barplot(tabScore, col=rgb(1,0,0))
```



Transparent

```
barplot(tabScore, col=rgb(1,0,0,.3))
```



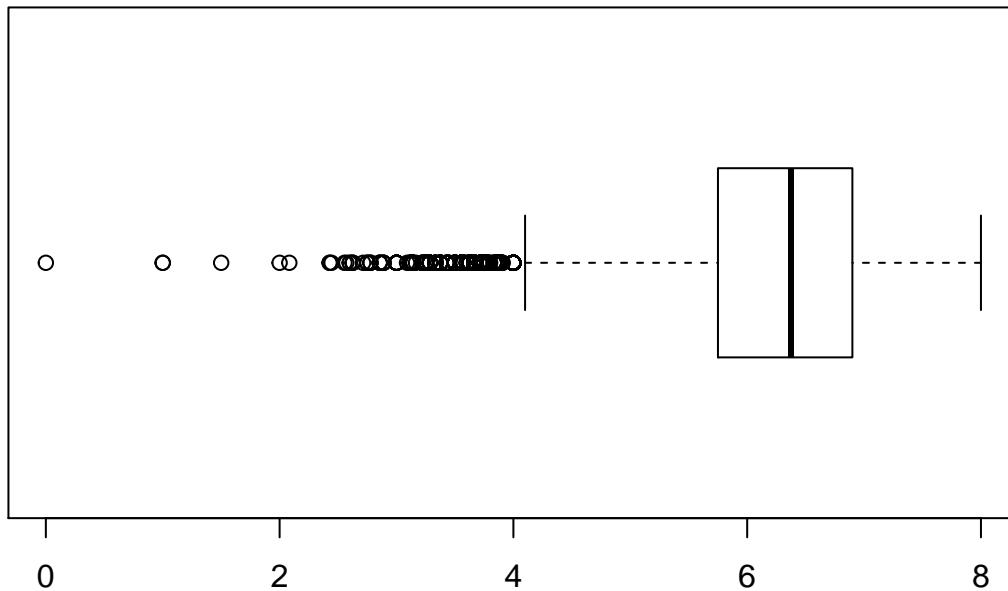
Boxplot

- Einen einfachen Boxplot erstellt man mit `boxplot()`
- Auch `boxplot()` muss mindestens ein Beobachtungsvektor übergeben werden

```
?boxplot
```

Horizontaler Boxplot

```
boxplot(Chem97$gcsescore,  
horizontal=TRUE)
```



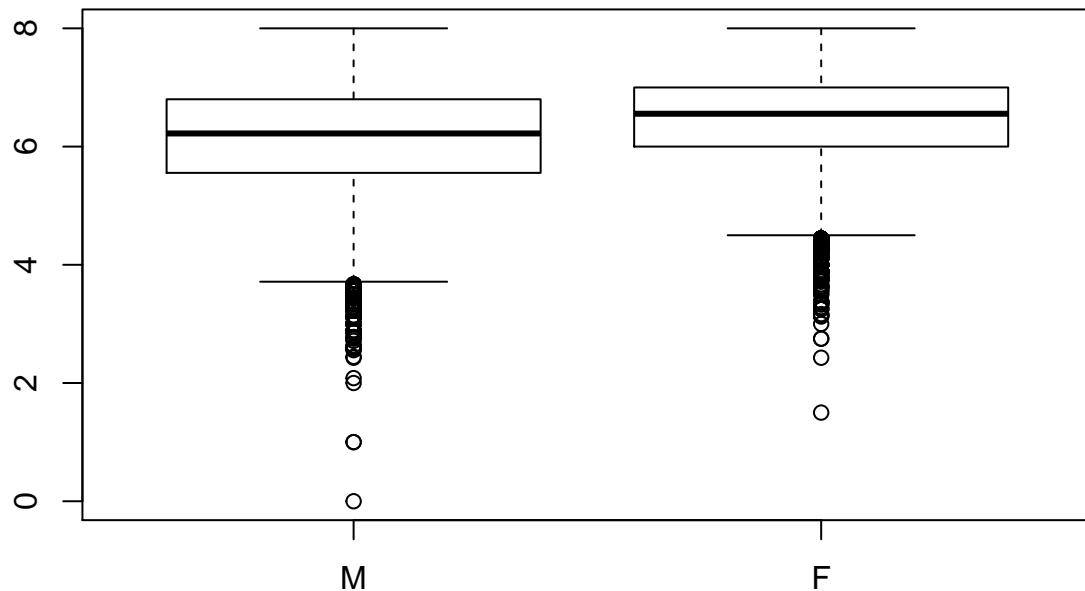
- Erklärung zu Boxplots

Gruppierte Boxplots

- Ein sehr einfacher Weg, einen ersten Eindruck über bedingte Verteilungen zu bekommen ist über sog. Gruppierte notched Boxplots
- Dazu muss der Funktion `boxplot()` ein sog. Formel-Objekt übergeben werden
- Die bedingende Variable steht dabei auf der rechten Seite einer Tilde

Beispiel grupperter Boxplot

```
boxplot(Chem97$gcsescore~Chem97$gender)
```



Alternativen zu Boxplot

Violinplot

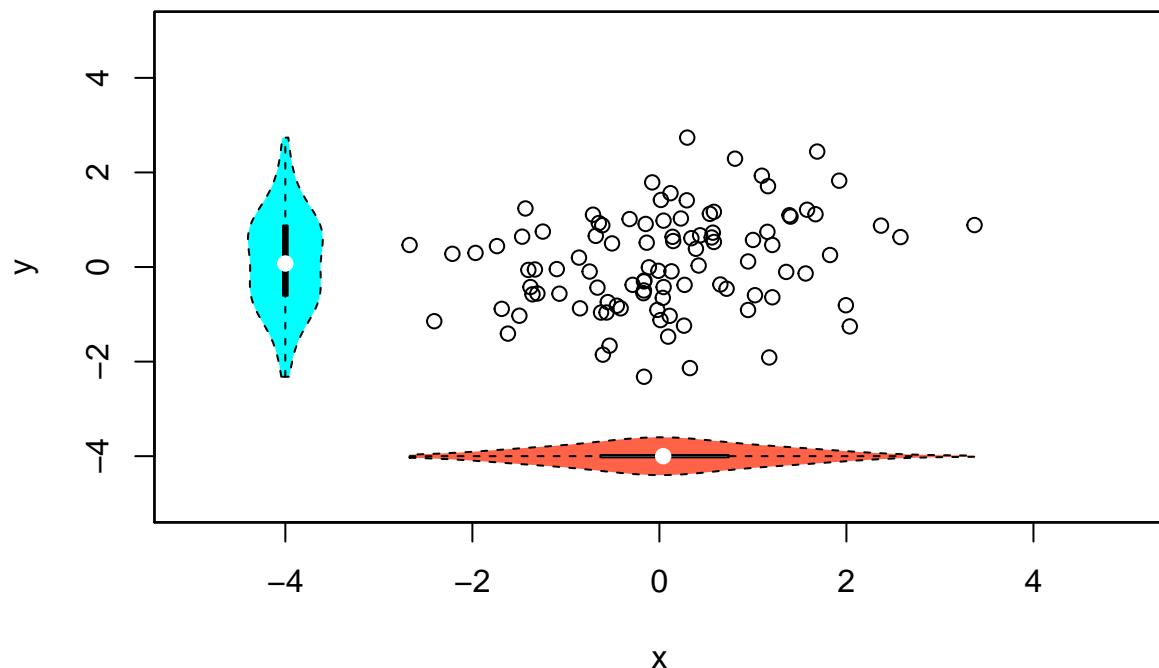
- Baut auf Boxplot auf
- Zusätzlich Informationen über Dichte der Daten
- Dichte wird über Kernel Methode berechnet.
- weißer Punkt - Median
- Je weiter die Ausdehnung, desto größer ist die Dichte an dieser Stelle.

```
# Beispieldaten erzeugen
x <- rnorm(100)
y <- rnorm(100)
```

Die Bibliothek vioplot

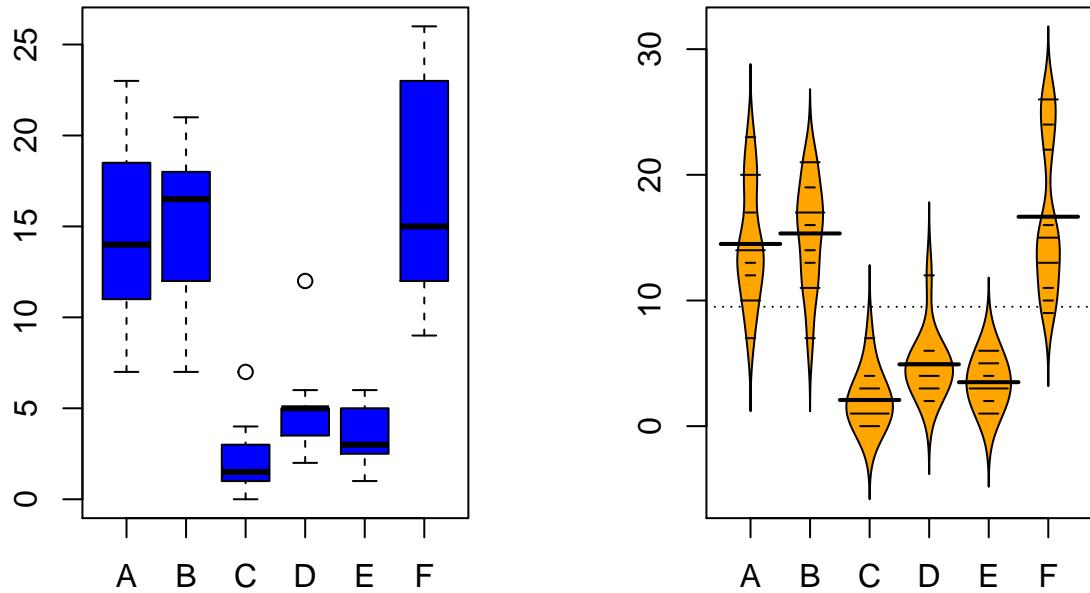
```
library(vioplot)
plot(x, y, xlim=c(-5,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-4,
        add=TRUE, lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=-4,
        add=TRUE, lty=2)
```

vioplot - Das Ergebnis



Alternativen zum Boxplot

```
library(beanplot)
par(mfrow = c(1,2))
boxplot(count~spray,data=InsectSprays,col="blue")
beanplot(count~spray,data=InsectSprays,col="orange")
```



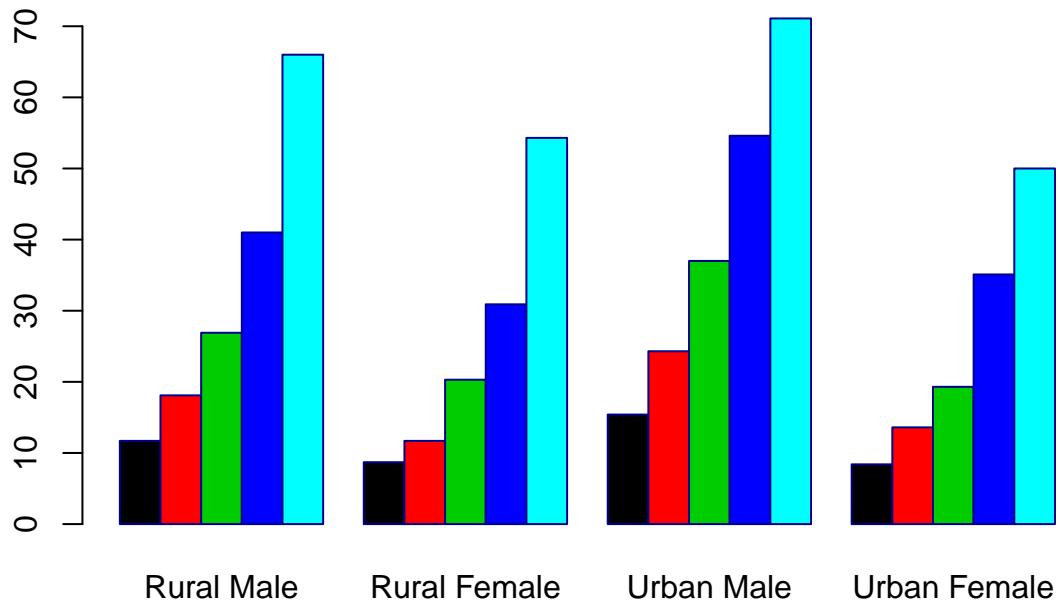
Grafiken für bedingte, bi- und multivariate Verteilungen

Scatterplots

- Ein einfacher two-way scatterplot kann mit der Funktion `plot()` erstellt werden
- `plot()` muss mindestens ein x und ein y Beobachtungsvektor übergeben werden
- Um die Farbe der Plot-Symbole anzupassen gibt es die Option `col` (Farbe als character oder numerisch)
- Die Plot-Symbole selbst können mit `pch` (plotting character) angepasst werden (character oder numerisch)
- Die Achsenbeschriftungen (`labels`) werden mit `xlab` und `ylab` definiert

Aufgabe - einfache Grafiken

- Laden Sie den Datensatz `VADeaths` und erzeugen Sie den folgenden plot:



Zurück zur Gliederung.

Zusammenhang

Edgar Anderson's Iris Daten

```

data(iris)
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1       3.5        1.4       0.2  setosa
## 2         4.9       3.0        1.4       0.2  setosa
## 3         4.7       3.2        1.3       0.2  setosa
## 4         4.6       3.1        1.5       0.2  setosa
## 5         5.0       3.6        1.4       0.2  setosa
## 6         5.4       3.9        1.7       0.4  setosa

```

petal length and width - Blütenblatt Länge und Breite

sepal length and width - Kelchblatt Länge und Breite

- Wikipedia Artikel zum IRIS Datensatz

Zusammenhang zwischen stetigen Variablen

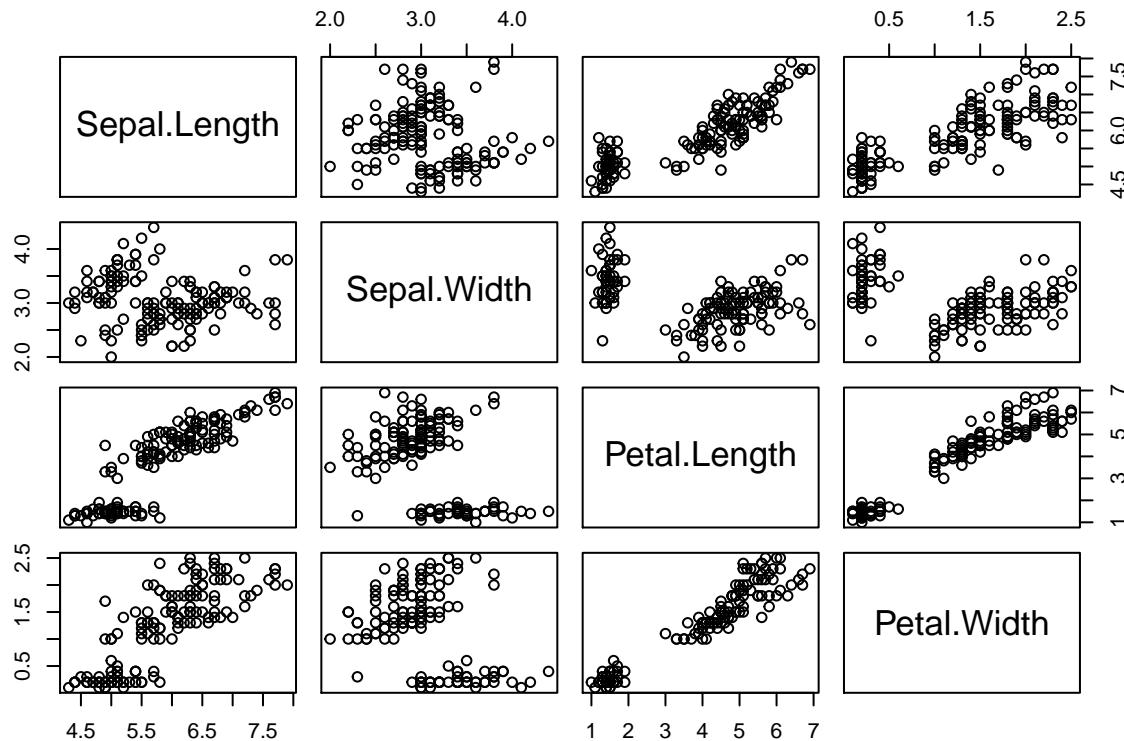
```
# Pearson Korrelationskoeffizient  
cor(iris$Sepal.Length,iris$Petal.Length)
```

```
## [1] 0.8717538
```

- Korrelation zwischen Länge Kelchblatt und Blütenblatt 0,87
- Der Pearson'sche Korrelationskoeffizient ist die default methode in cor().

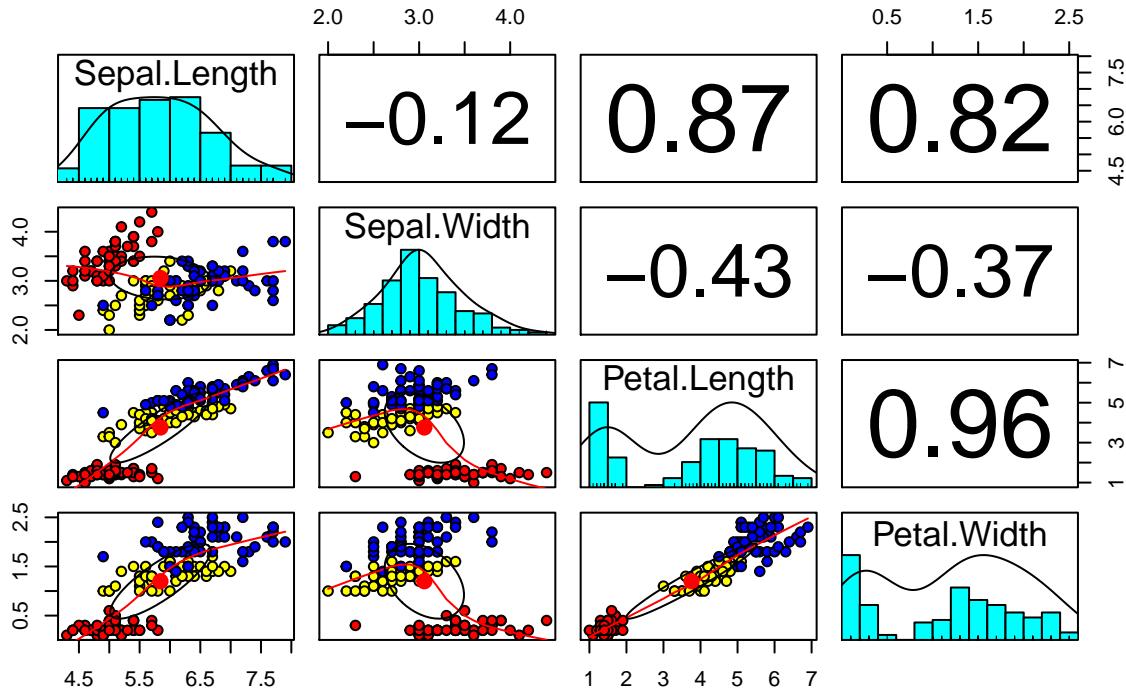
Zusammenhang zwischen mehreren Variablen

```
pairs(iris[,1:4])
```



Zusammenhang zwischen mehreren Variablen

```
library("psych")  
pairs.panels(iris[1:4],bg=c("red","yellow","blue")  
[iris$Species],pch=21,main="")
```



Verschiedene Korrelationskoeffizienten

```
# Pearson Korrelationskoeffizient
cor(iris[,1:4])

##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000 -0.1175698   0.8717538   0.8179411
## Sepal.Width     -0.1175698  1.0000000  -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401   1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259   0.9628654   1.0000000

# Kendall's tau (Rangkorrelation)
cor(iris[,1:4], method = "kendall")

##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000 -0.07699679   0.7185159   0.6553086
## Sepal.Width     -0.07699679  1.00000000  -0.1859944  -0.1571257
## Petal.Length    0.71851593 -0.18599442   1.0000000   0.8068907
## Petal.Width     0.65530856 -0.15712566   0.8068907   1.0000000

# Spearman's rho (Rangkorrelation)
cor(iris[,1:4], method = "spearman")

##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000 -0.1667777   0.8818981   0.8342888
## Sepal.Width     -0.1667777  1.0000000  -0.3096351  -0.2890317
```

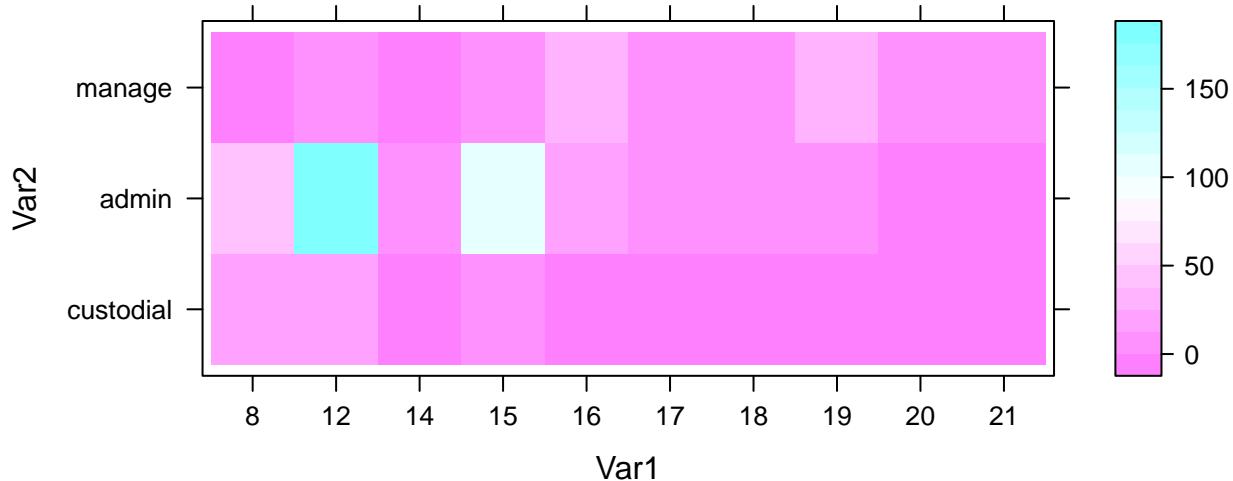
```
## Petal.Length    0.8818981 -0.3096351    1.0000000  0.9376668
## Petal.Width     0.8342888 -0.2890317    0.9376668  1.0000000
```

Zusammenhang zwischen kategorialen Variablen

- chisq.test() testet, ob zwei kategoriale Merkmale stochastisch unabhängig sind.
- Getestet wird gegen die Nullhypothese der Gleichverteilung

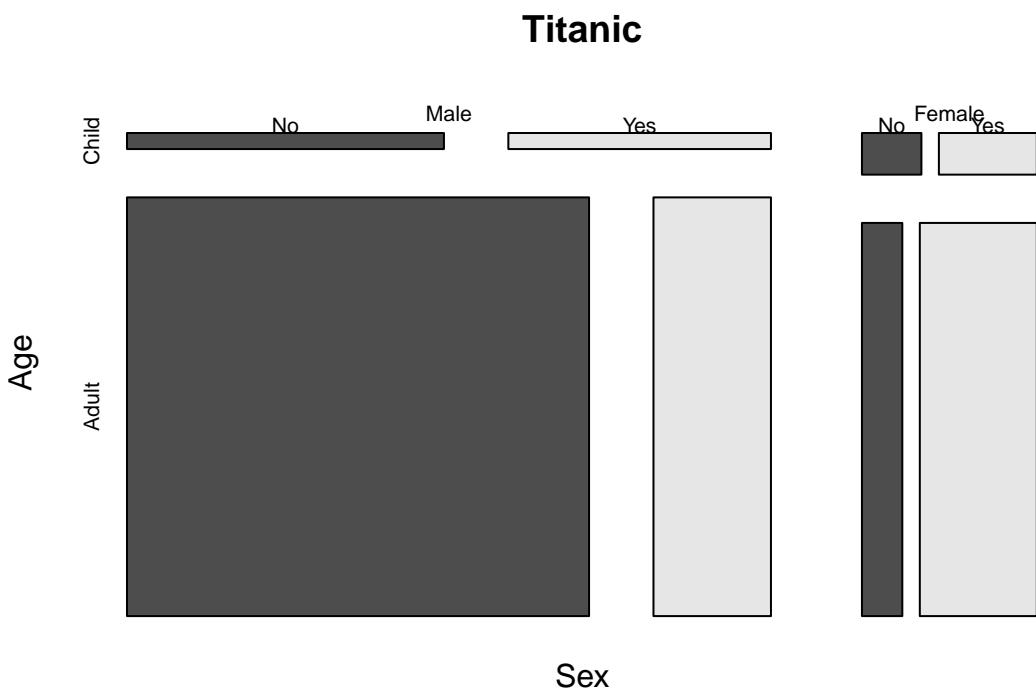
Levelplot

```
library("lattice")
library("AER")
data(BankWages)
levelplot(table(BankWages$education,BankWages$job))
```



Visualisierung von Zusammenhängen zwischen kategorialen Variablen

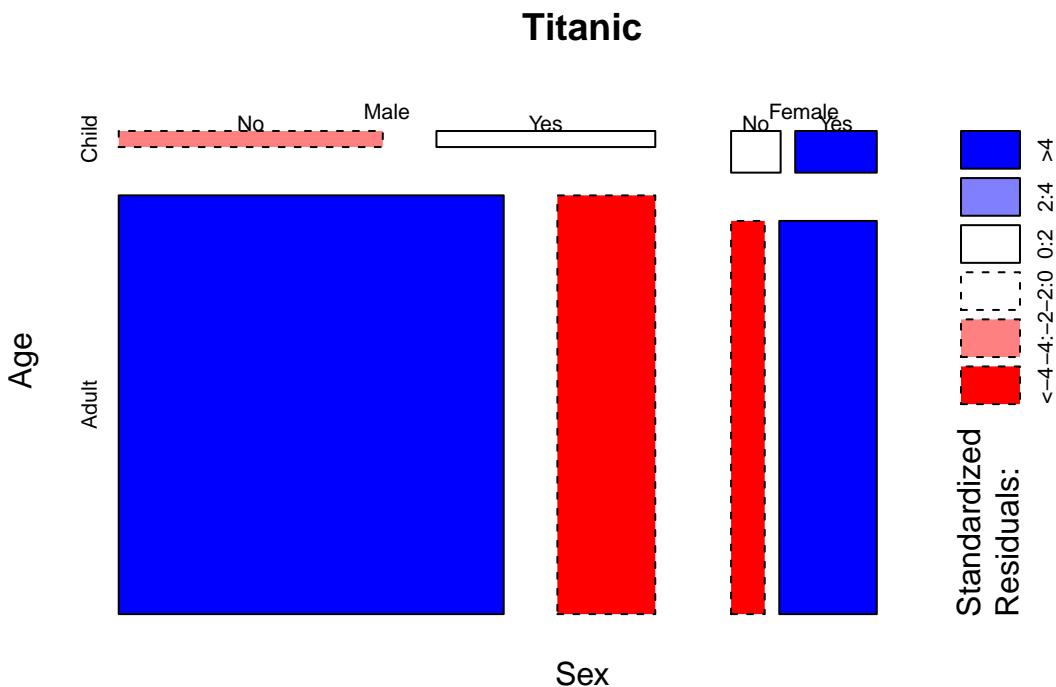
```
mosaicplot(~ Sex + Age + Survived,
           data = Titanic, color = TRUE)
```



Shading

Flächen werden entsprechend der Residuen eingefärbt:

```
mosaicplot(~ Sex + Age + Survived,
           data = Titanic, shade = TRUE)
```



Literatur zu Zusammenhangsmaßen

- Methodensammlung mit R
- Beispiele zu Zusammenhangsmaßen
- Umsetzung in R

Sachs - Angewandte Statistik mit R

Das lattice Paket

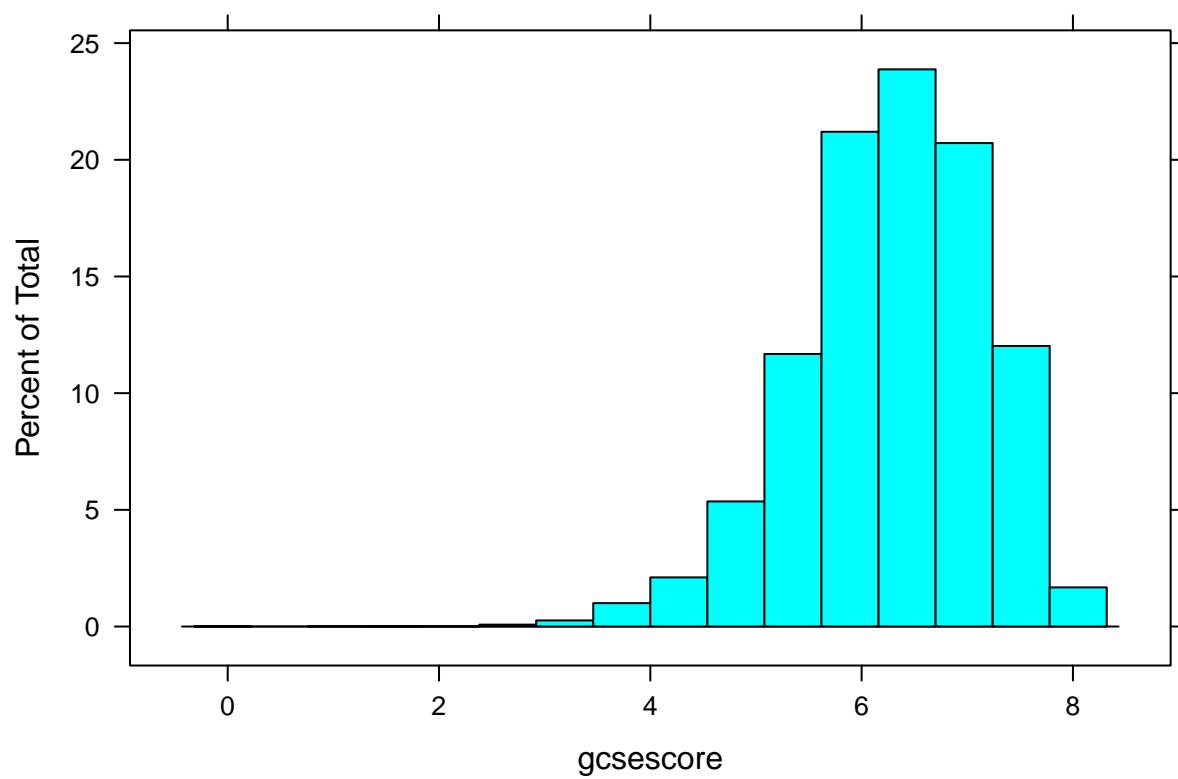
Das lattice-Paket

It is designed to meet most typical graphics needs with minimal tuning, but can also be easily extended to handle most nonstandard requirements.

<http://stat.ethz.ch/R-manual/R-devel/library/lattice/html/Lattice.html>

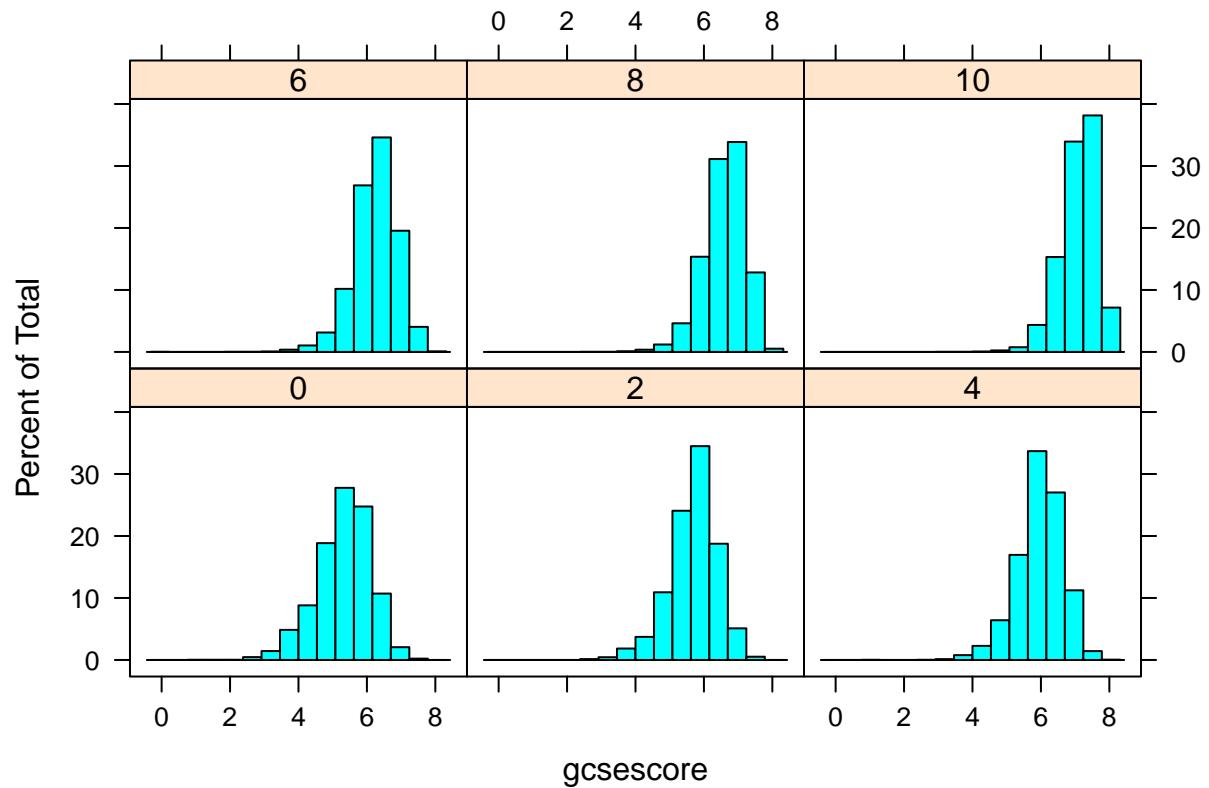
Histogramm mit Lattice

```
library("lattice");library("mlmRev")
data(Chem97)
histogram(~ gcsescore, data = Chem97)
```



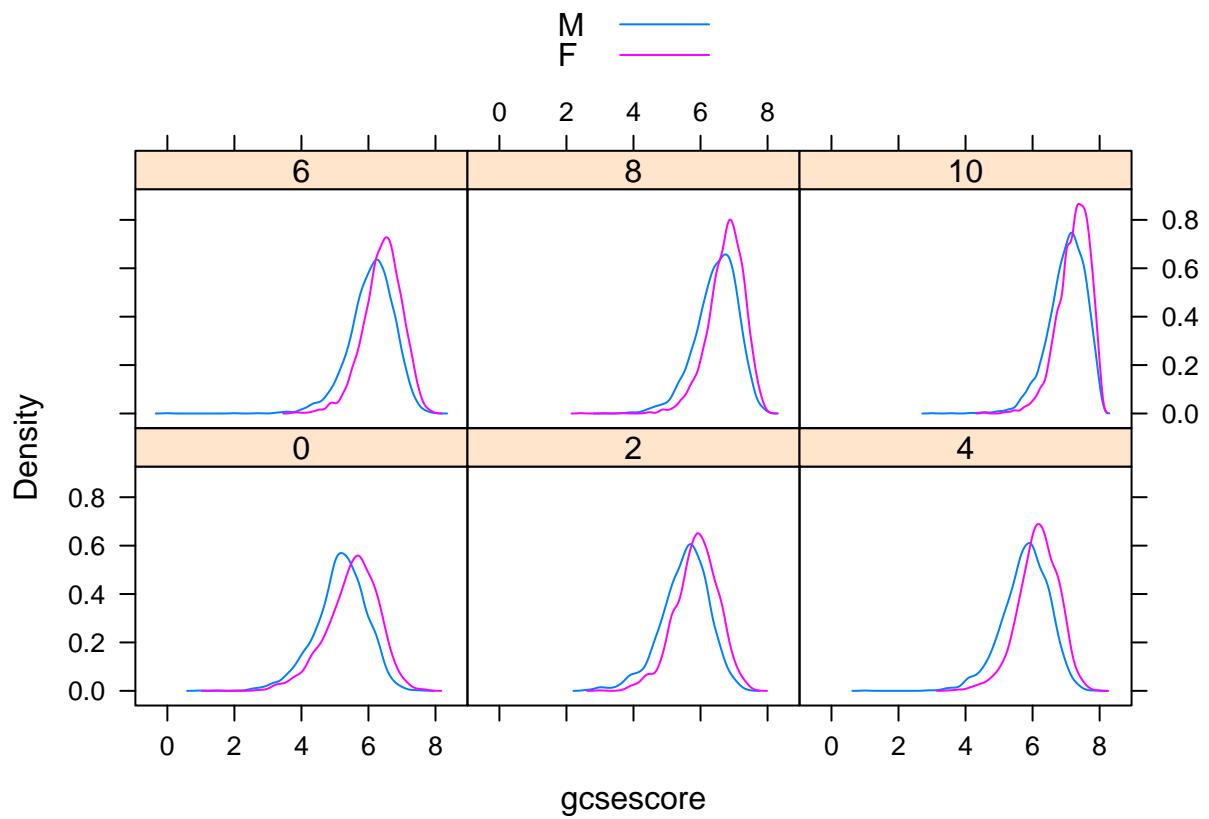
Histogramm mit Lattice

```
histogram(~ gcse score | factor(score), data = Chem97)
```



Die Dichte mit Lattice zeichnen

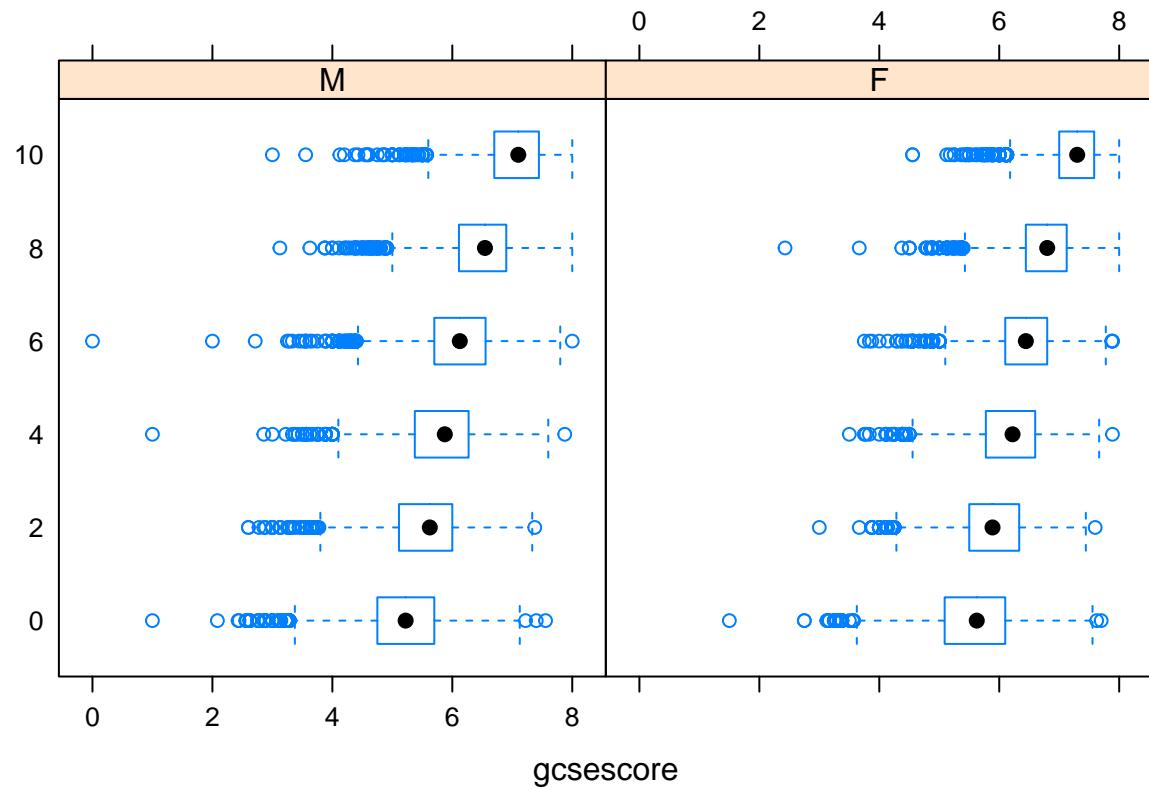
```
densityplot(~ gcsescore | factor(score), Chem97,
           groups=gender, plot.points=FALSE, auto.key=TRUE)
```



Einführung in das Paket lattice

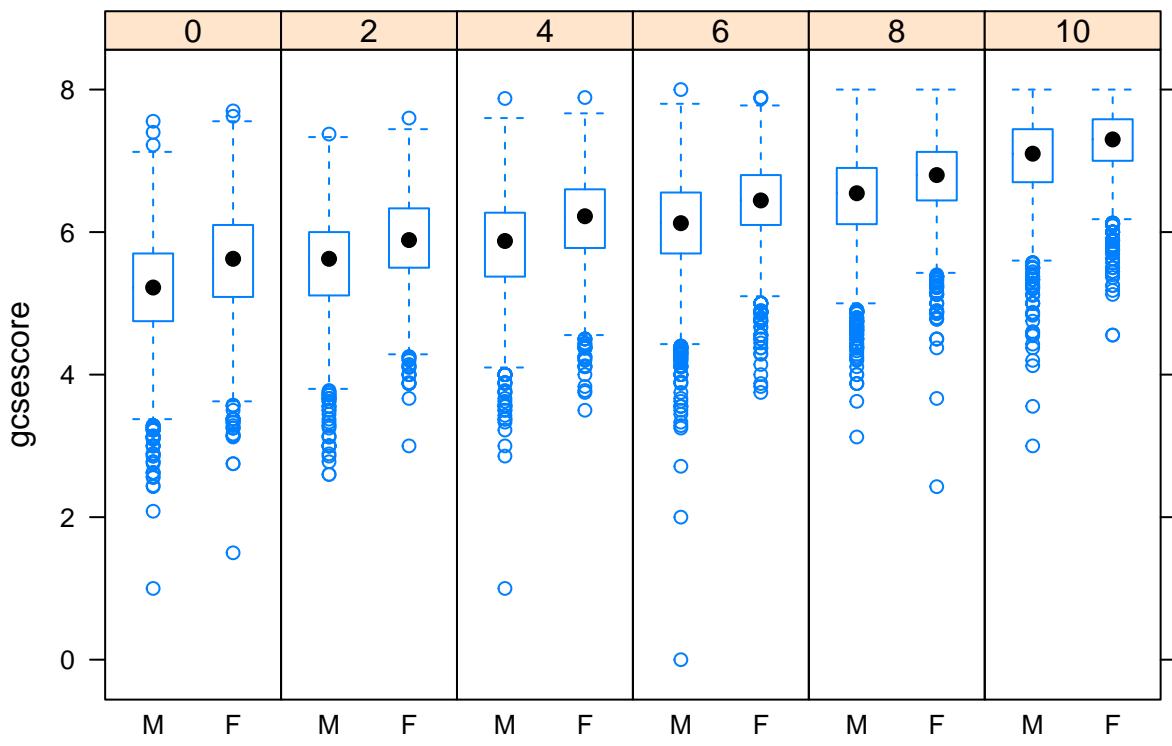
Boxplot mit Lattice zeichnen

```
bwplot(factor(score) ~ gcsescore | gender, Chem97)
```



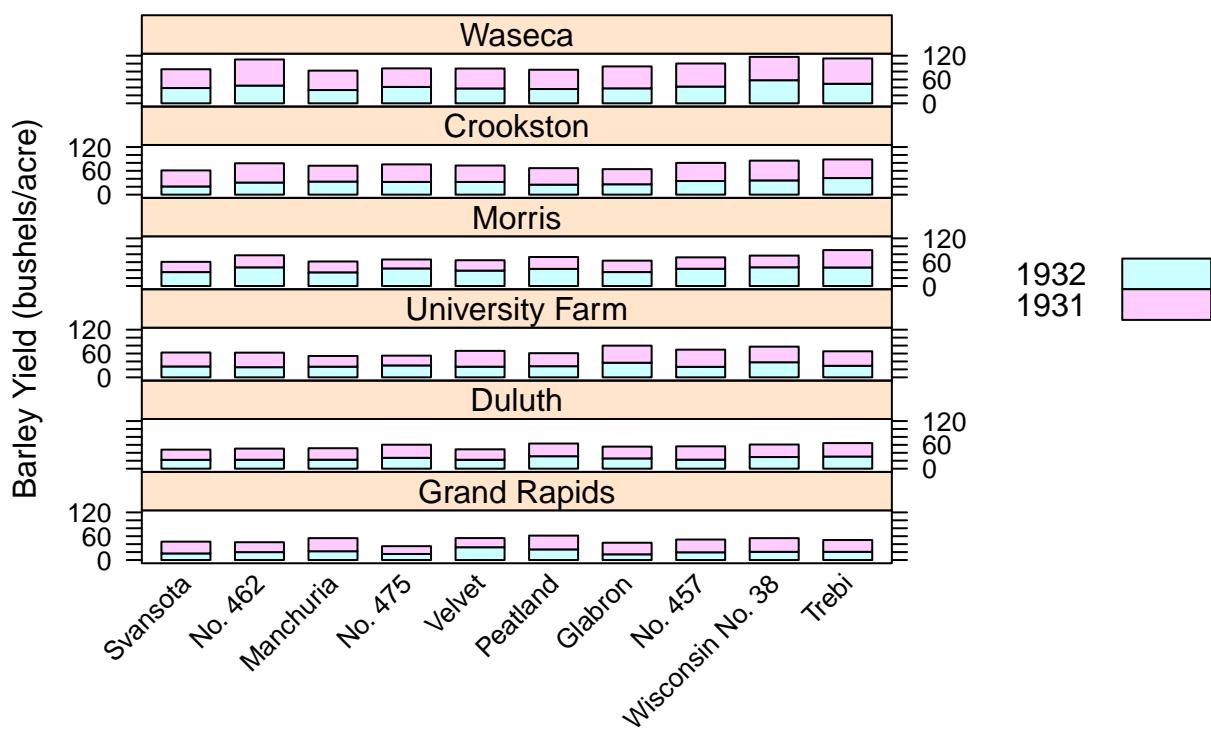
Boxplot mit Lattice zeichnen

```
bwplot(gcsescore ~ gender | factor(score), Chem97,  
       layout = c(6, 1))
```



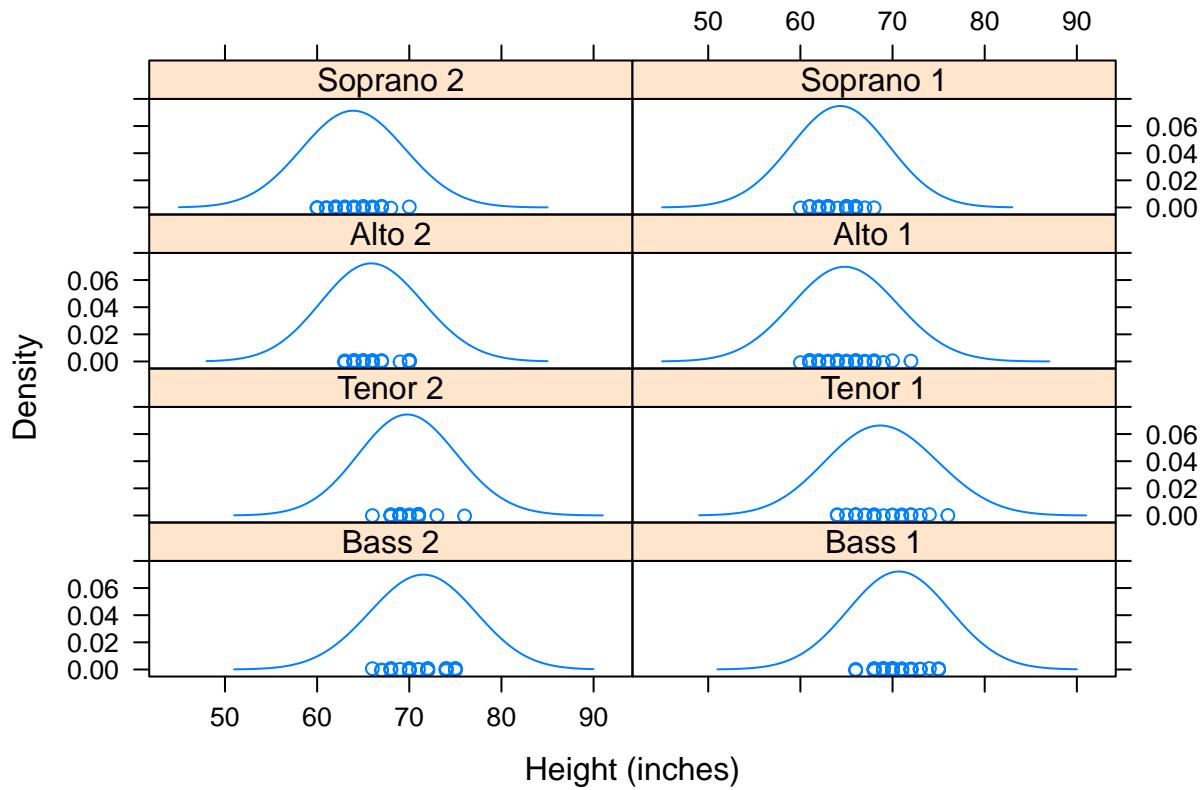
Univariate Plots

```
barchart(yield ~ variety | site, data = barley,
         groups = year, layout = c(1,6), stack = TRUE,
         auto.key = list(space = "right"),
         ylab = "Barley Yield (bushels/acre)",
         scales = list(x = list(rot = 45)))
```



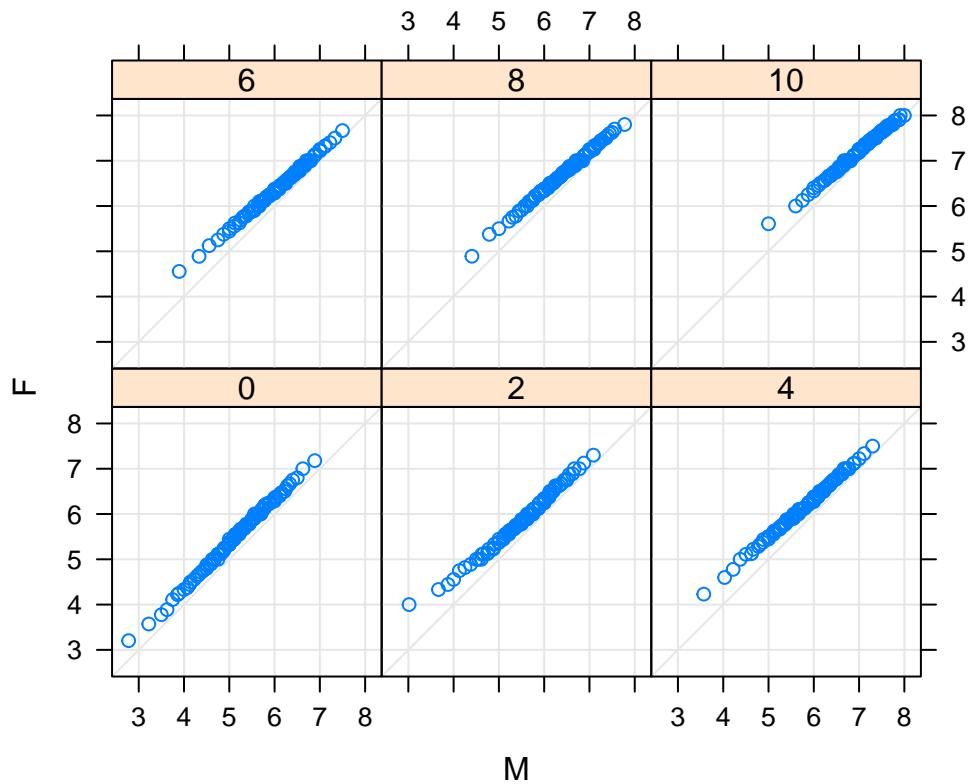
Densityplot

```
densityplot(~ height | voice.part, data = singer, layout = c(2, 4),
           xlab = "Height (inches)", bw = 5)
```



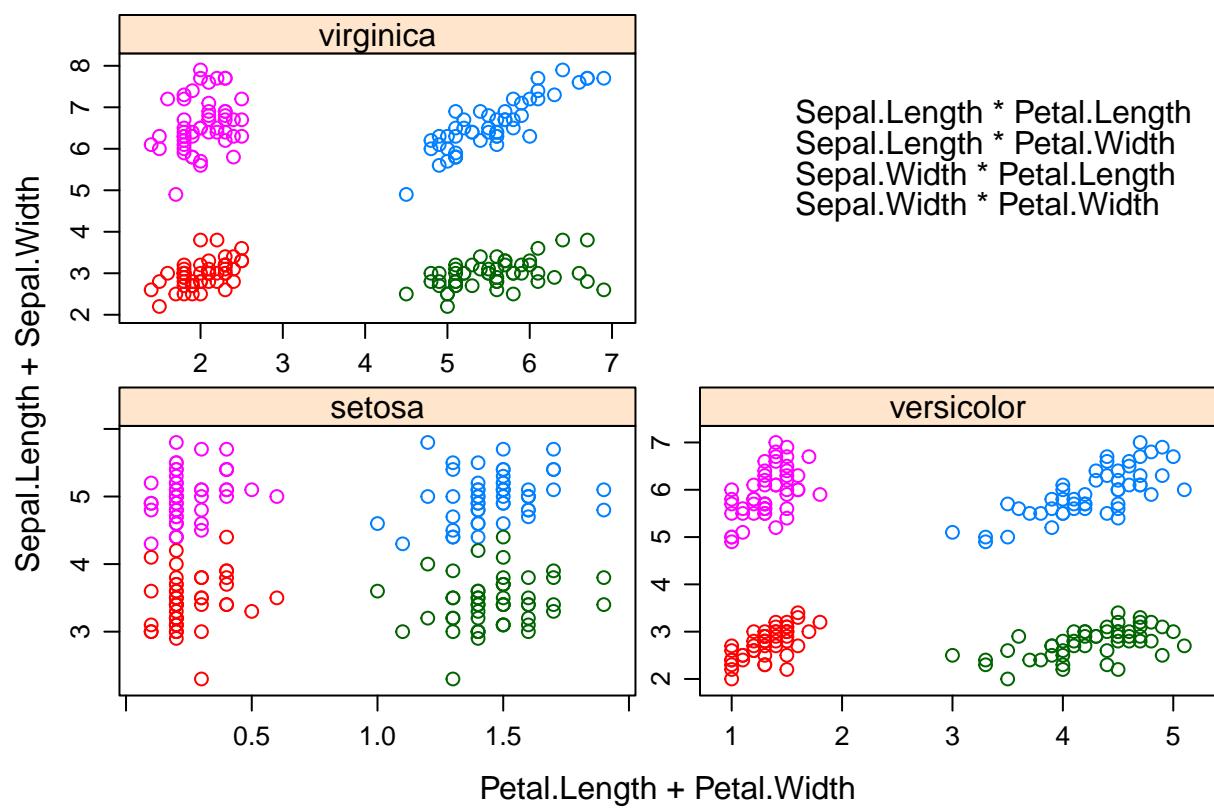
Bivariate Plots

```
qq(gender ~ gcsescore | factor(score), Chem97,
  f.value = ppoints(100), type = c("p", "g"), aspect = 1)
```



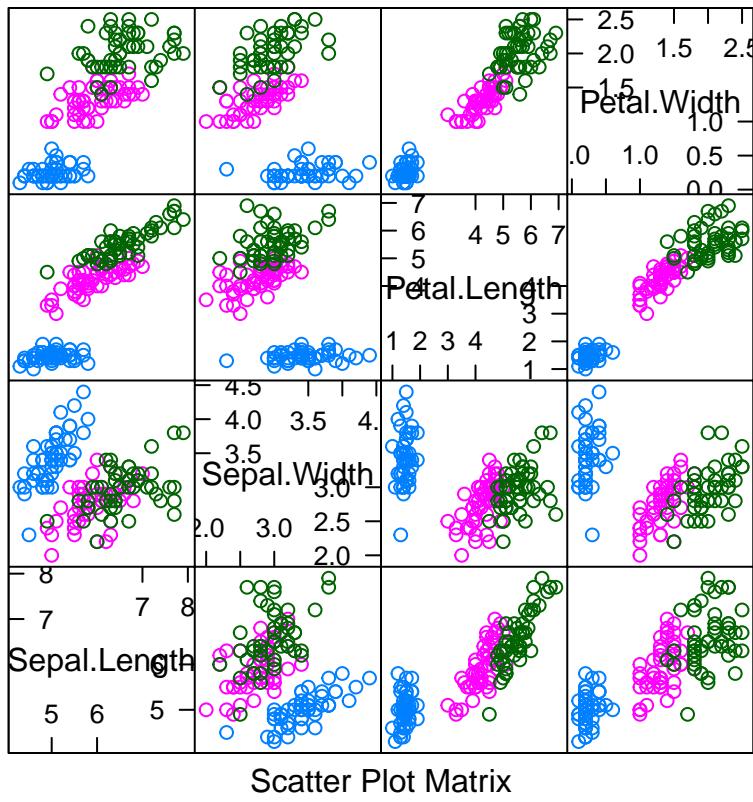
xypplot

```
xypplot(Sepal.Length + Sepal.Width ~ Petal.Length + Petal.Width | Species,
        data = iris, scales = "free", layout = c(2, 2),
        auto.key = list(x = .6, y = .7, corner = c(0, 0)))
```



Multivariate Plots

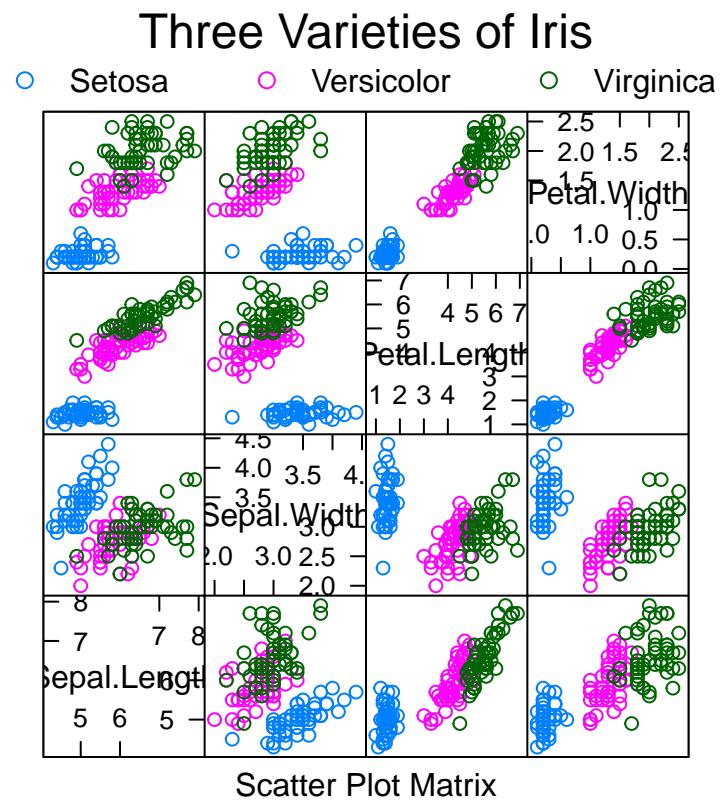
```
splom(~iris[1:4], groups = Species, data = iris)
```



Scatter Plot Matrix

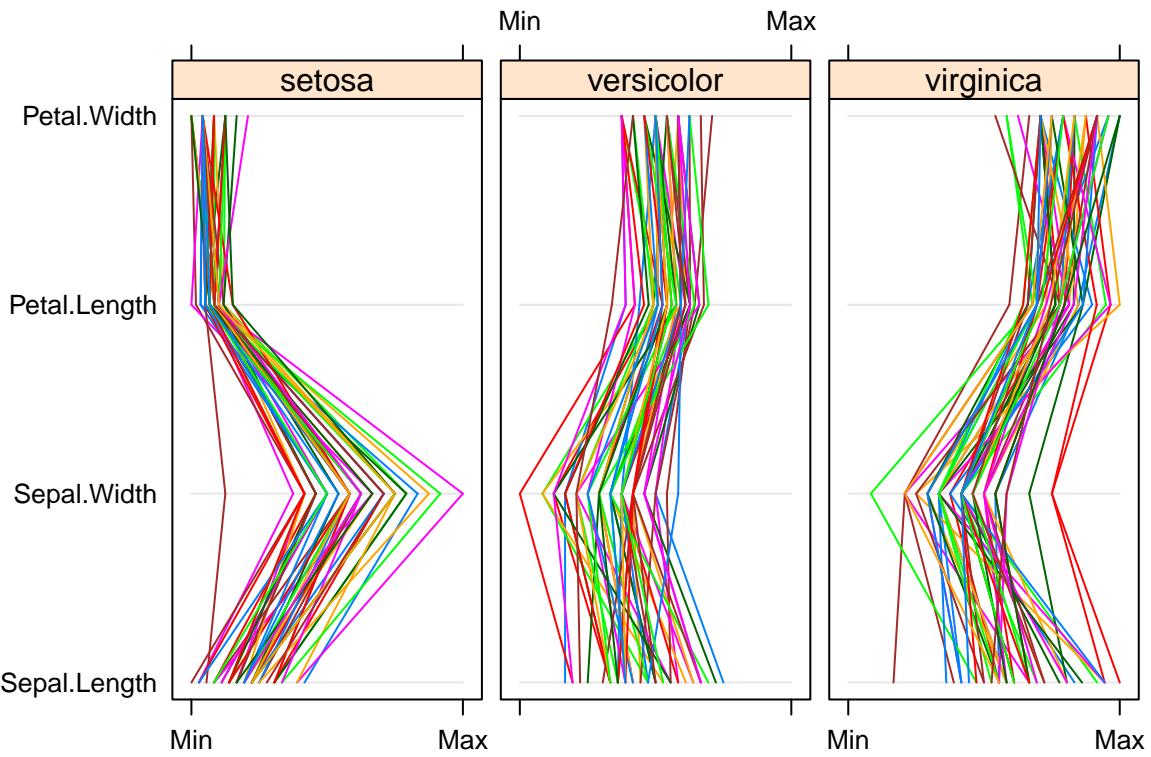
```

super.sym <- trellis.par.get("superpose.symbol")
splom(~iris[1:4], groups = Species, data = iris,
      panel = panel.superpose,
      key = list(title = "Three Varieties of Iris",
                 columns = 3,
                 points = list(pch = super.sym$pch[1:3],
                               col = super.sym$col[1:3]),
                 text = list(c("Setosa", "Versicolor", "Virginica"))))
```



`parallelplot`

```
parallelplot(~iris[1:4] | Species, iris)
```



Lattice Befehle

- Übersicht aller Lattice Befehle

Aufgabe - OECD Datensatz

- Laden Sie den oecd-Datensatz herunter und lesen Sie ihn mit folgender Funktion ein:

```
data <- read.csv("oecd.csv", header = TRUE)
```

- Überprüfen Sie die Dimension der OECD-Daten.
- Berechnen Sie die Mittelwerte und Varianzen der einzelnen Variablen mit einem geeigneten apply Befehl.
- In welchem Land waren die meisten Jugendlichen mindestens zweimal betrunken? Wie hoch ist der maximale Prozentsatz?
- In welchem Land ist die Sterblichkeit am geringsten? Wie hoch ist sie in diesem Land?
- Erstellen Sie einen neuen Datensatz, der aufsteigend nach dem Einkommen geordnet ist. Speichern Sie diesen in einer neuen .csv Datei

Zurück zur Gliederung.

Die lineare Regression

Die lineare Regression

Maindonald - DataAnalysis

- Einführung in R
- Datenanalyse
- Statistische Modelle
- Inferenzkonzepte
- Regression mit einem Prädiktor
- Multiple lineare Regression
- Ausweitung des linearen Modells
- ...

Lineare Regression in R - Beispieldatensatz

John H. Maindonald and W. John Braun

DAAG - Data Analysis and Graphics Data and Functions

```
install.packages("DAAG")
```

```
library("DAAG")
data(roller)
```

help on roller data:

```
?roller
```

Das lineare Regressionsmodell in R

Schätzen eines Regressionsmodells:

```
roller.lm <- lm(depression ~ weight, data = roller)
```

So bekommt man die Schätzwerte:

```
summary(roller.lm)
```

```
##
## Call:
## lm(formula = depression ~ weight, data = roller)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -8.180 -5.580 -1.346  5.920  8.020 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.0871     4.7543  -0.439  0.67227  
## weight       2.6667     0.7002   3.808  0.00518 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.735 on 8 degrees of freedom
```

```

## Multiple R-squared:  0.6445, Adjusted R-squared:  0.6001
## F-statistic:  14.5 on 1 and 8 DF,  p-value: 0.005175

```

Falls das Modell ohne Intercept geschätzt werden soll:

```
lm(depression ~ -1 + weight, data = roller)
```

```

##
## Call:
## lm(formula = depression ~ -1 + weight, data = roller)
##
## Coefficients:
## weight
## 2.392

```

Summary des Modells

```

summary(roller.lm)

##
## Call:
## lm(formula = depression ~ weight, data = roller)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -8.180 -5.580 -1.346  5.920  8.020 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.0871    4.7543  -0.439  0.67227    
## weight       2.6667    0.7002   3.808  0.00518 **  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 6.735 on 8 degrees of freedom
## Multiple R-squared:  0.6445, Adjusted R-squared:  0.6001 
## F-statistic:  14.5 on 1 and 8 DF,  p-value: 0.005175

```

R arbeitet mit Objekten

- `roller.lm` ist nun ein spezielles Regressions-Objekt
- Auf dieses Objekt können nun verschiedene Funktionen angewendet werden

```
predict(roller.lm) # Vorhersage
```

```

##      1        2        3        4        5        6        7
## 2.979669 6.179765 6.713114 10.713233 12.046606 14.180002 14.980026
##      8        9       10
## 18.180121 24.046962 30.980502

```

```
resid(roller.lm) # Residuen
```

```

##      1        2        3        4        5        6
## -0.9796695 -5.1797646 -1.7131138 -5.7132327  7.9533944  5.8199976
##      7        8        9       10

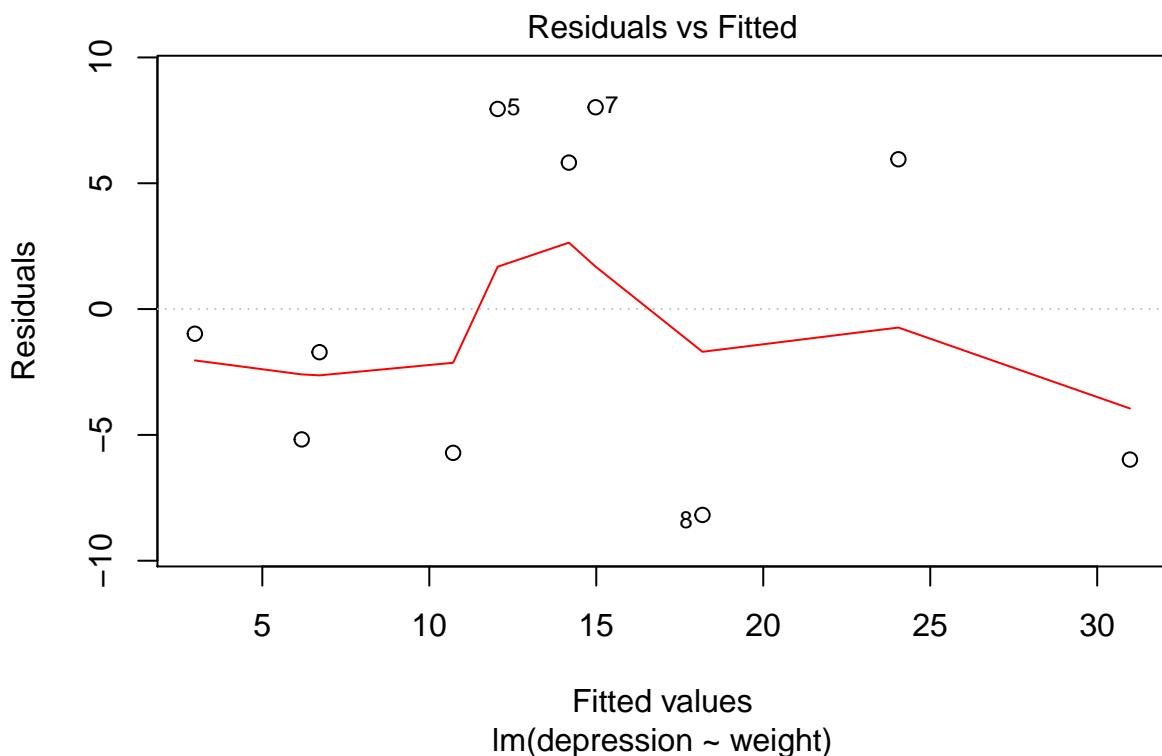
```

```
## 8.0199738 -8.1801213 5.9530377 -5.9805017
```

Residuenplot

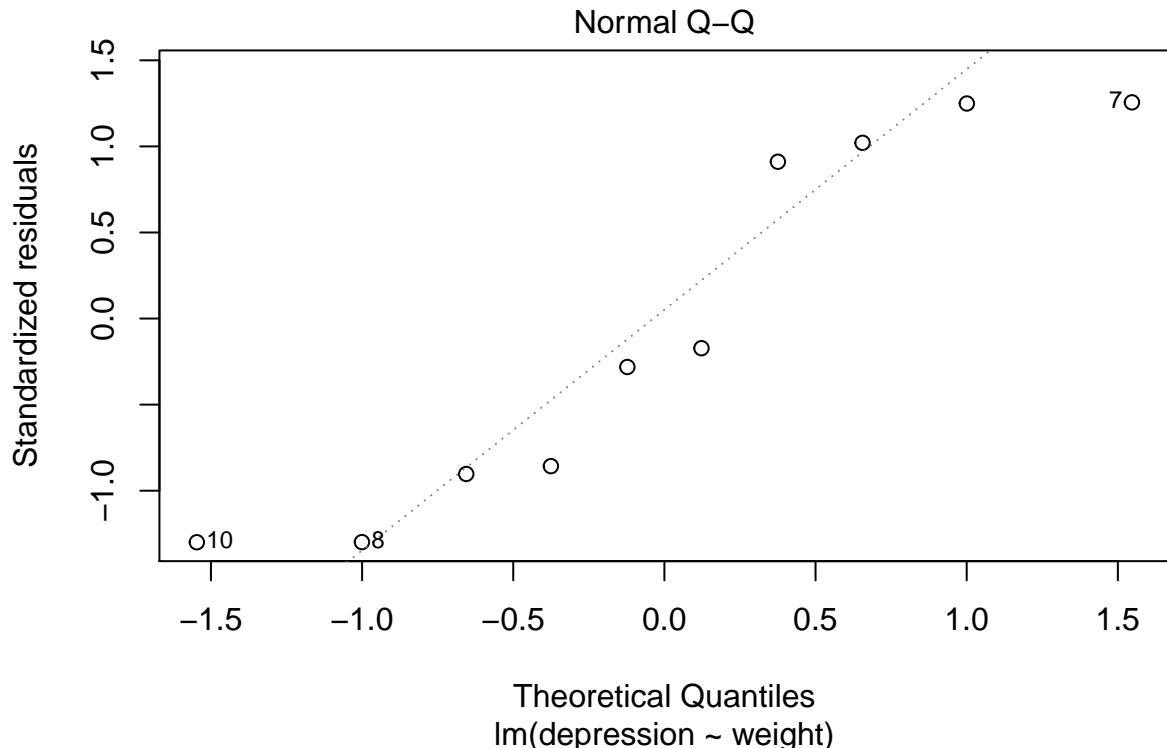
- Sind Annahmen des linearen Regressionsmodells verletzt?
- Dies ist der Fall, wenn ein Muster abweichend von einer Linie zu erkennen ist.
- Hier ist der Datensatz sehr klein

```
plot(roller.lm,1)
```



Residuenplot

```
plot(roller.lm,2)
```



- Wenn die Residuen normalverteilt sind sollten sie auf einer Linie liegen.

Linkliste - lineare Regression

- Regression - r-bloggers
- Das Komplette Buch von Faraway- sehr intuitiv geschrieben.
- Gute Einführung auf Quick-R
- Multiple Regression

Aufgabe - lineare Regression

Beschrieben wird Wegstrecke, dreier Spielzeugautos die in unterschiedlichen Winkeln Rampe herunterfuhren.

- angle: Winkel der Rampe
- distance: Zurückgelegte Strecke des Spielzeugautos
- car: Autotyp (1, 2 oder 3)

- (a) Lesen Sie den Datensatz `toycars` in einen dataframe `data` ein und wandeln Sie die Variable `car` des Datensatzes in einen Faktor (`as.factor`) um.
- (b) Erstellen Sie drei Boxplots, die die zurückgelegte Strecke getrennt nach dem Faktor `car` darstellen.
- (c) Schätzen Sie für jedes der 3 Autos separat die Parameter des folgenden linearen Modells mit Hilfe der Funktion `lm()`

$$distance_i = \beta_0 + \beta_1 \cdot angle_i + \epsilon_i$$

- (d) Überprüfen Sie deskriptiv die Anpassung der drei Modelle, indem Sie die Regressioner- ade in einen Plot von distance gegen angle einfügen. Deutet das

$$R^2$$

jeweils auf eine gute Modellanpassung hin?

Zurück zur Gliederung.

Die logistische Regression

Agresti - Categorical Data Analysis (2002)

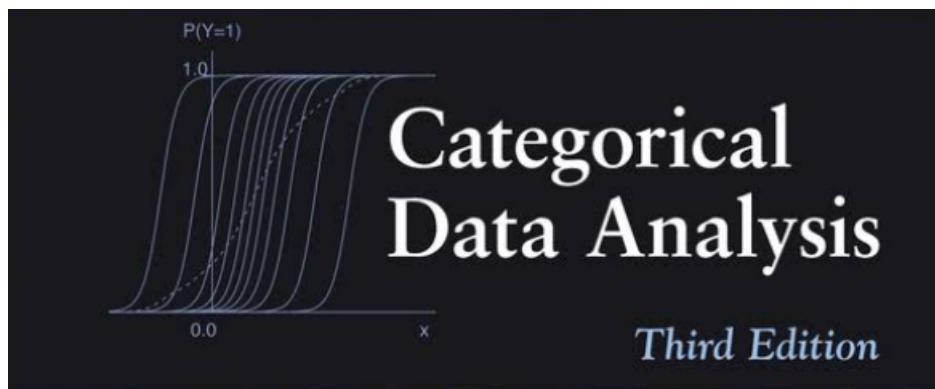


Figure 26:

- Sehr intuitiv geschriebenes Buch
- Sehr ausführliches begleitendes Skript von Thompson
- Das Skript eignet sich um die kategoriale Datenanalyse nachzuvollziehen

Faraway Bücher zu Regression in R

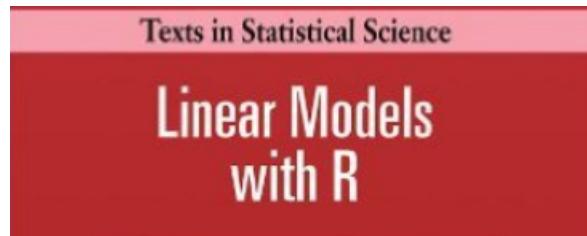


Figure 27:

- Logistische Regressionen gut erklärt
- Beispiele mit R-code
 - Faraway - Extending the linear model with r
 - Faraway - Practical Regression and Anova using R

Binäre AVs mit `glm`

- Die logistische Regression gehört zur Klasse der generalisierten linearen Modelle (GLM)
- Die Funktion zur Schätzung eines Modells dieser Klasse heißt `glm()`
- `glm()` muss 1. ein Formel-Objekt mitgegeben werden und 2. die Klasse (binomial, gaussian, Gamma) samt link-Funktion (logit, probit, cauchit, log, cloglog)

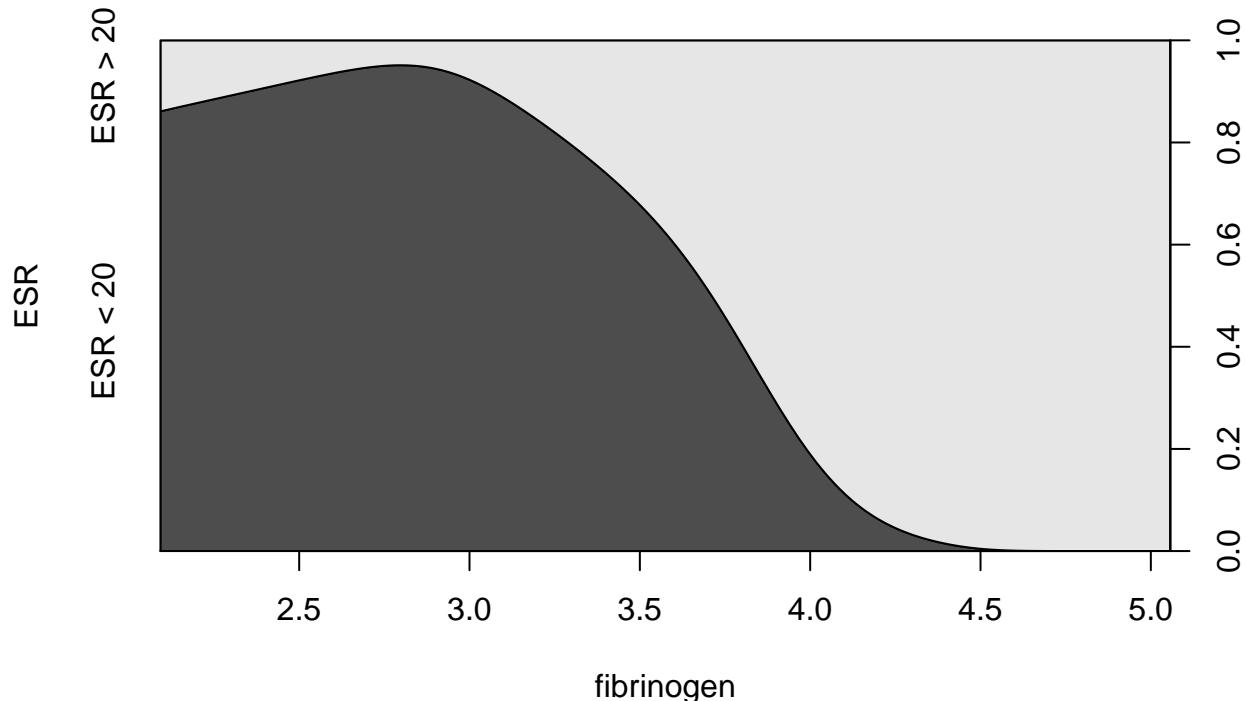
Beispieldaten für die logistische Regression

```
install.packages("HSAUR")
library("HSAUR")
data("plasma", package = "HSAUR")
```

Logistische Regression mit R

- Kategoriale Daten:

```
cdplot(ESR ~ fibrinogen, data = plasma)
```



Logistische Regression mit R

```
plasma_glm_1 <- glm(ESR ~ fibrinogen, data = plasma,
                      family = binomial())
```

Weitere Beispieldaten

```
install.packages("faraway")
library("faraway")
data(orings)
```

temp	damage
53	5
57	1
58	1

Generalisierte Regression mit R - weitere Funktionen

- Logistisches Modell mit Probit-Link:

```
probitmod <- glm(cbind(damage, 6-damage) ~ temp,
                  family=binomial(link=probit), orings)
```

- Regression mit Zähldaten:

```
modp <- glm(Species ~ ., family=poisson, gala)
```

- Proportional odds logistic regression im Paket MASS:

```
library("MASS")
house.plr<-polr(Sat~Infl, weights=Freq, data=housing)
```

Linkliste - logistische Regression

- Einführung in logistische Regression



Figure 28:

- Code zum Buch von Faraway

```

library(faraway)
data(orings)
plot(damage/6 ~ temp, orings, xlim=c(25,85), ylim = c(0,1), xlab="Temperature", ylab="Prob of damage")
lmod <- lm(damage/6 ~ temp, orings)
abline(lmod)
logitmod <- glm(cbind(damage, 6-damage) ~ temp, family=binomial, orings)
summary(logitmod)
plot(damage/6 ~ temp, orings, xlim=c(25,85), ylim = c(0,1), xlab="Temperature", ylab="Prob of damage")

```

Figure 29:

Aufgabe - Datenanalyse

- Laden Sie einen Datensatz Ihrer Wahl - entweder einen eigenen oder einen der vorgestellten Datensätze
- Berechnen Sie einfache Statistiken auf den wichtigsten Variablen (Mittelwert, Median, Standardabweichung)
- Erzeugen Sie eine zweidimensionale Häufigkeitstabelle
- Führen Sie eine Regression mit sinnvoll gewählten abhängiger und unabhängiger Variablen auf den Daten durch
- Erzeugen Sie einen Lattice-plot

Zurück zur Gliederung.

Faktoren in R

Was sind Faktoren in R

- Faktoren können eine begrenzte Zahl von Ausprägungen annehmen
- Sie werden oft auch als kategoriale Variablen bezeichnet
- Sie sind vor allem bei der Modellierung wichtig
- Faktoren werden anders behandelt als stetige Variablen
- Wenn man diese Variablen richtig definiert werden sie auch von R richtig behandelt

<http://www.stat.berkeley.edu/~s133/factors.html>

Beispiel Definition von Faktoren

```

data <- c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata <- factor(data)
fdata

## [1] 1 2 2 3 1 2 3 3 1 2 3 3 1
## Levels: 1 2 3

• labels direkt definieren

rdata <- factor(data,labels=c("I","II","III"))
rdata

## [1] I   II  III I   II  III III I   II  III III I
## Levels: I II III

```

Weitere Möglichkeit der Definition

```
levels(fdata) <- c('I','II','III')
fdata

## [1] I   II  II  III I   II  III III I   II  III III I
## Levels: I II III
```

Beispiel Monate

```
mons <- c("March","April","January",
        "November","January","September",
        "October","September","November",
        "August","January","November",
        "November","February","May",
        "August","July","December",
        "August","August","September",
        "November","February","April")
mons <- factor(mons)
table(mons)

## mons
##      April     August December February January       July March
##         2          4         1         2         3         1         1
##      May November October September
##         1          5         1         3
```

Beispiel: ordered factor

```
mons <- factor(mons,levels=c("January","February",
                             "March","April","May","June",
                             "July","August","September",
                             "October","November",
                             "December"),
                ordered=TRUE)

mons[1] < mons[2]

## [1] TRUE
```

Rücktransformation

```
fert <- c(10,20,20,50,10,20,10,50,20)
fert <- factor(fert,levels=c(10,20,50),ordered=TRUE)
fert

## [1] 10 20 20 50 10 20 10 50 20
## Levels: 10 < 20 < 50

mean(fert)

## [1] NA
```

```
mean(as.numeric(levels(fert)[fert]))
```

```
## [1] 23.33333
```

Tabellen mit Faktoren

```
lets <- sample(letters, size=100, replace=TRUE)
lets <- factor(lets)
table(lets[1:5])

##
## a b c d e f g h i k l m n o p q r s t u v w x y z
## 0 0 0 0 0 1 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

Grafiken mit ggplot

Das Paket ggplot2

- Entwickelt von Hadley Wickham
- Viele Informationen unter:
- <http://ggplot2.org/>
- Den Graphiken liegt eine eigene Grammatik zu Grunde

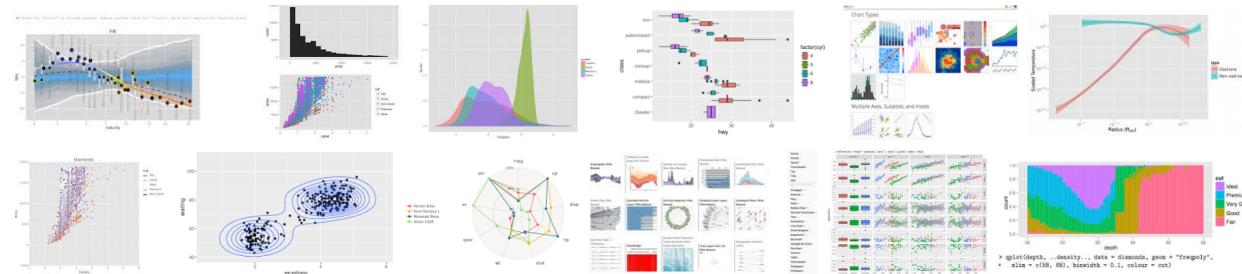


Figure 30:

Basiseinführung ggplot2

```
<www.r-bloggers.com/basic-introduction-to-ggplot2/>
```

```
install.packages("ggplot2")
```

```
library(ggplot2)
```

Der diamonds Datensatz

```
head(diamonds)
```

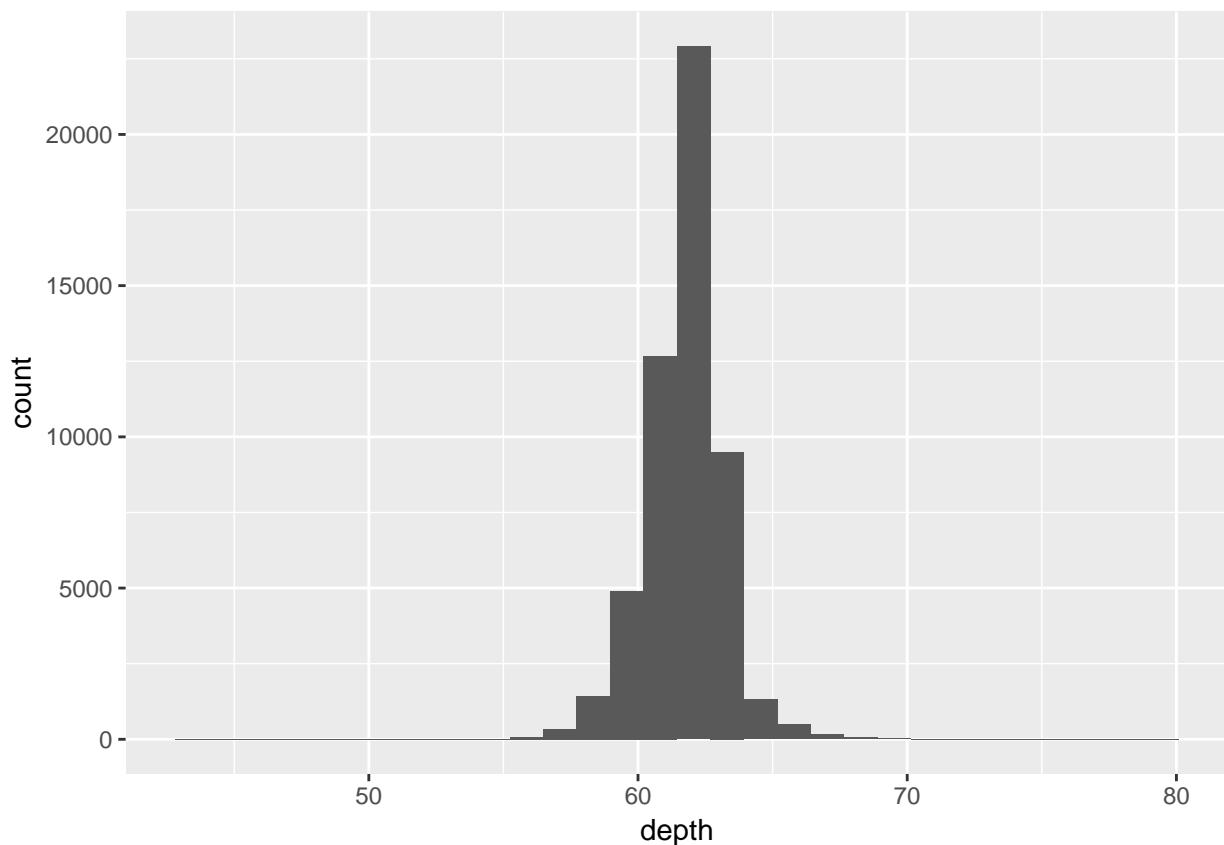
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43

carat	cut	color	clarity	depth	table	price	x	y	z
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

Wie nutzt man qplot

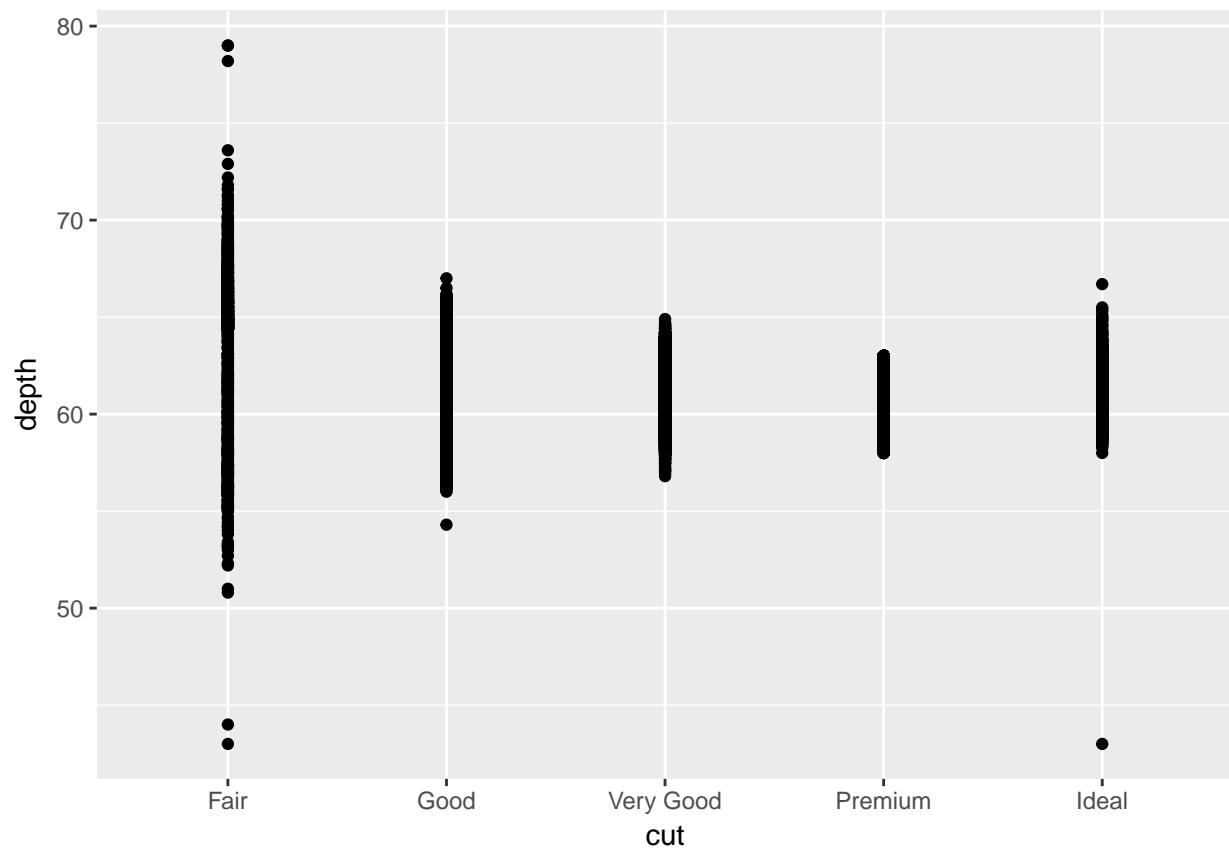
- Die Funktion `qplot` wird für schnelle Graphiken verwendet (quick plots)
- bei der Funktion `ggplot` kann man alles bis ins Detail kontrollieren

```
# histogram
qplot(depth, data=diamonds)
```



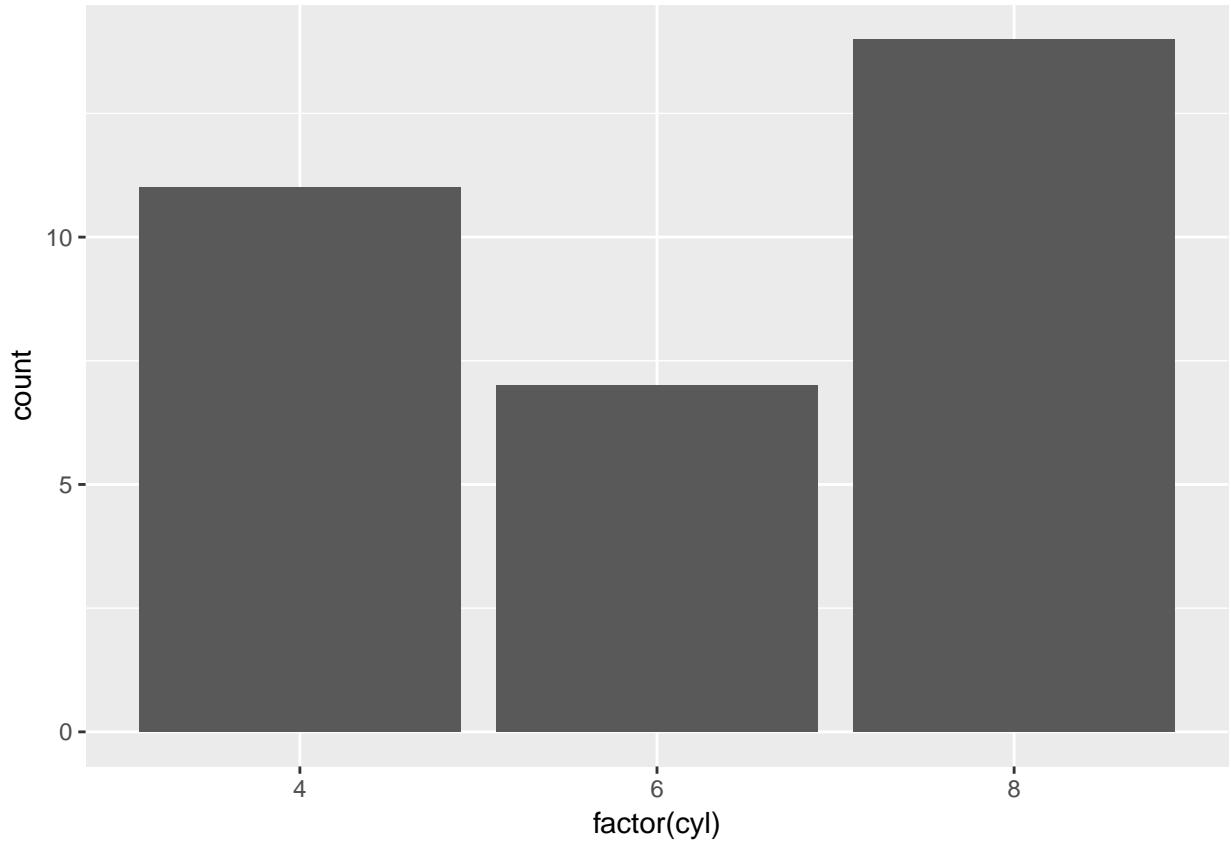
Ein Balkendiagramm

```
qplot(cut, depth, data=diamonds)
```



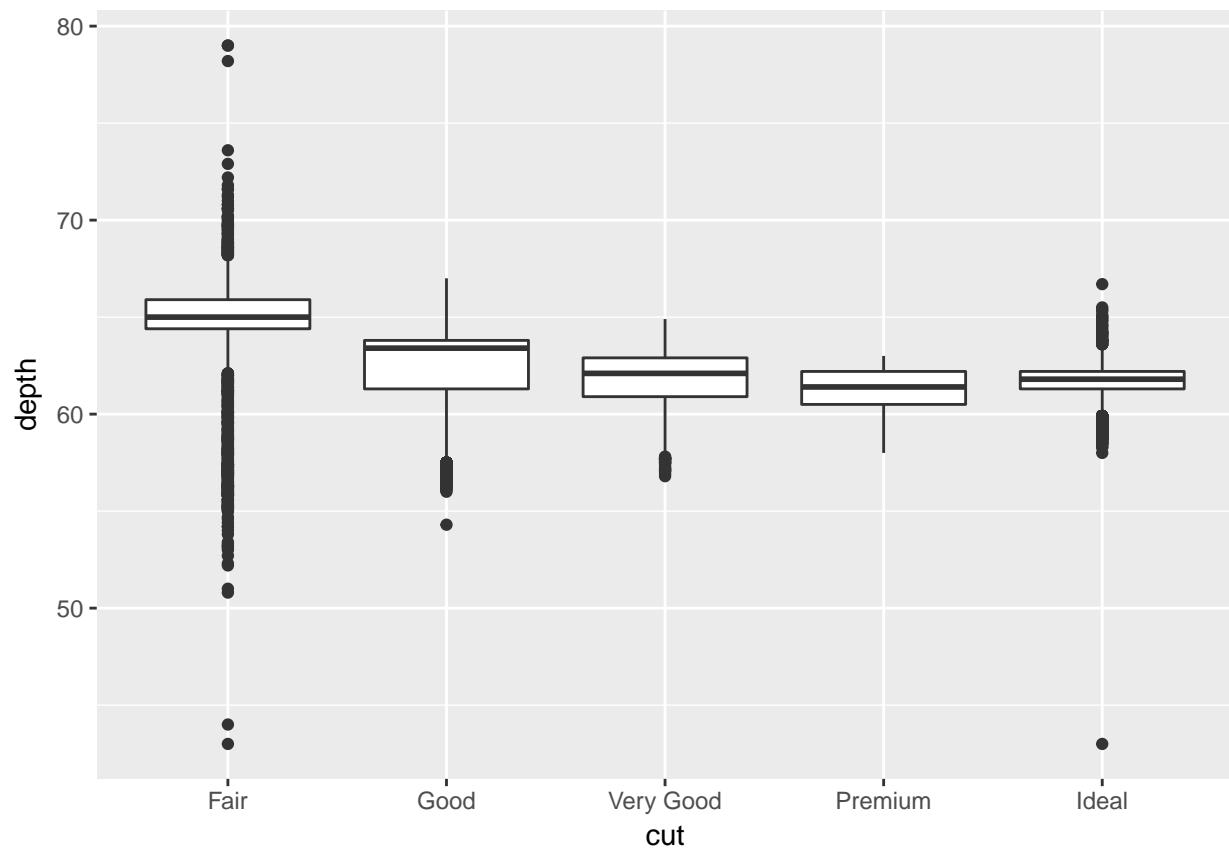
Ein weiteres Balkendiagramm

```
qplot(factor(cyl), data=mtcars, geom="bar")
```



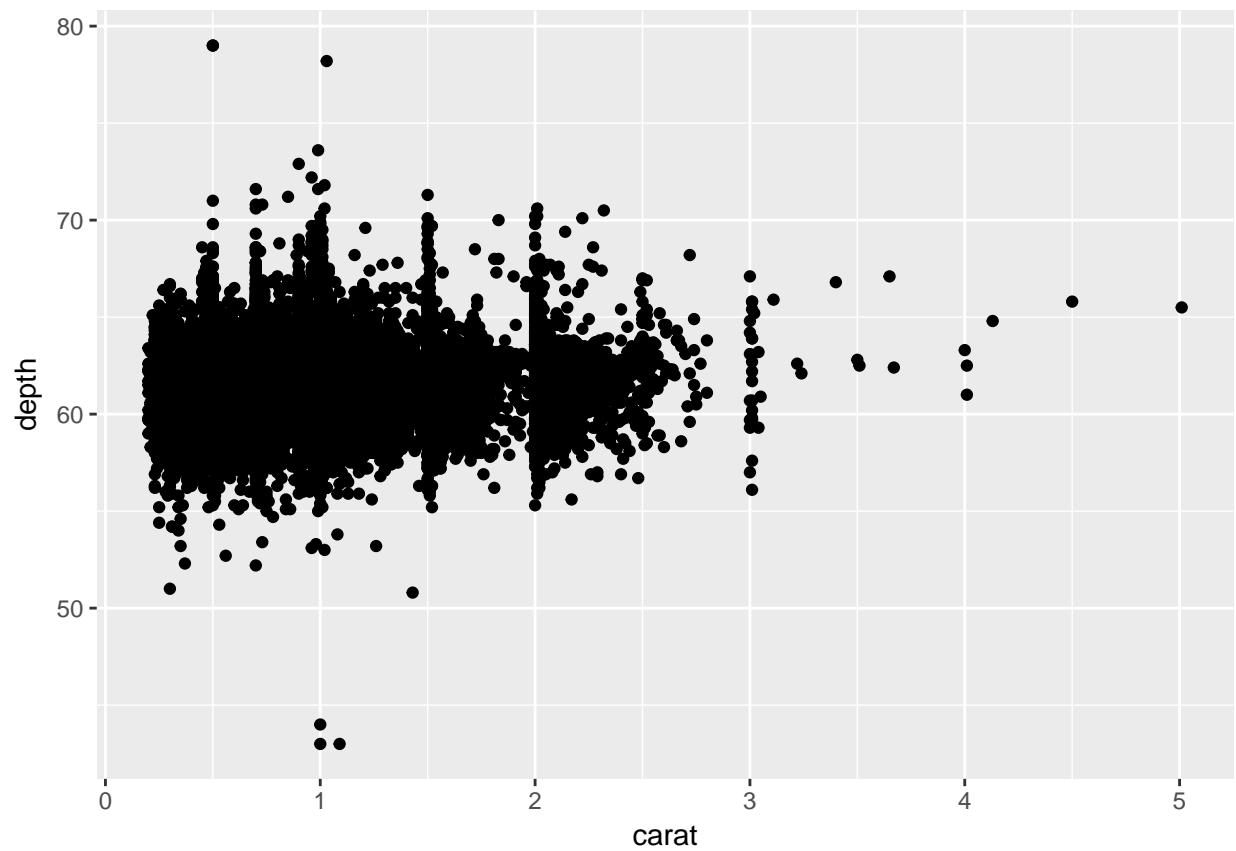
Boxplot

```
qplot(data=diamonds,x=cut,y=depth,geom="boxplot")
```



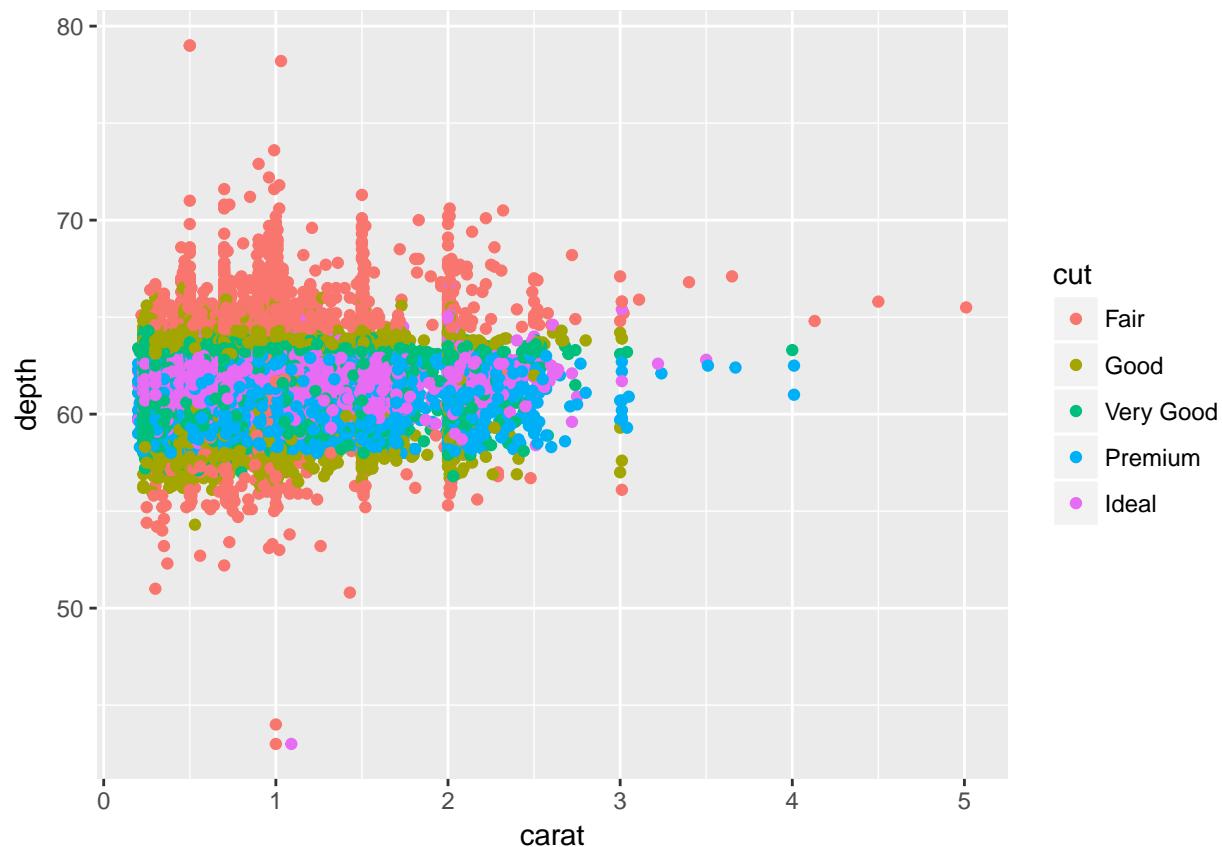
Scatterplot

```
# scatterplot
qplot(carat, depth, data=diamonds)
```



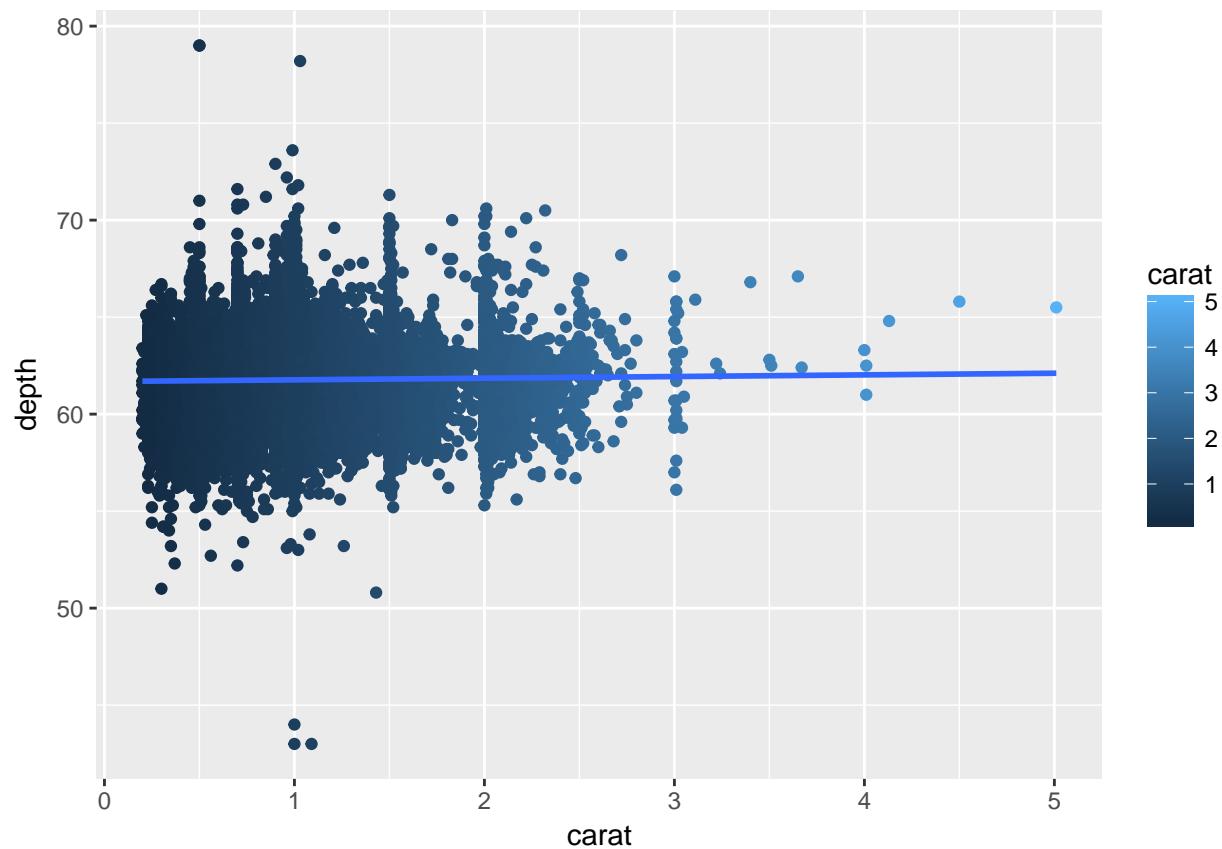
Farbe hinzufügen:

```
qplot(carat, depth, data=diamonds, color=cut)
```



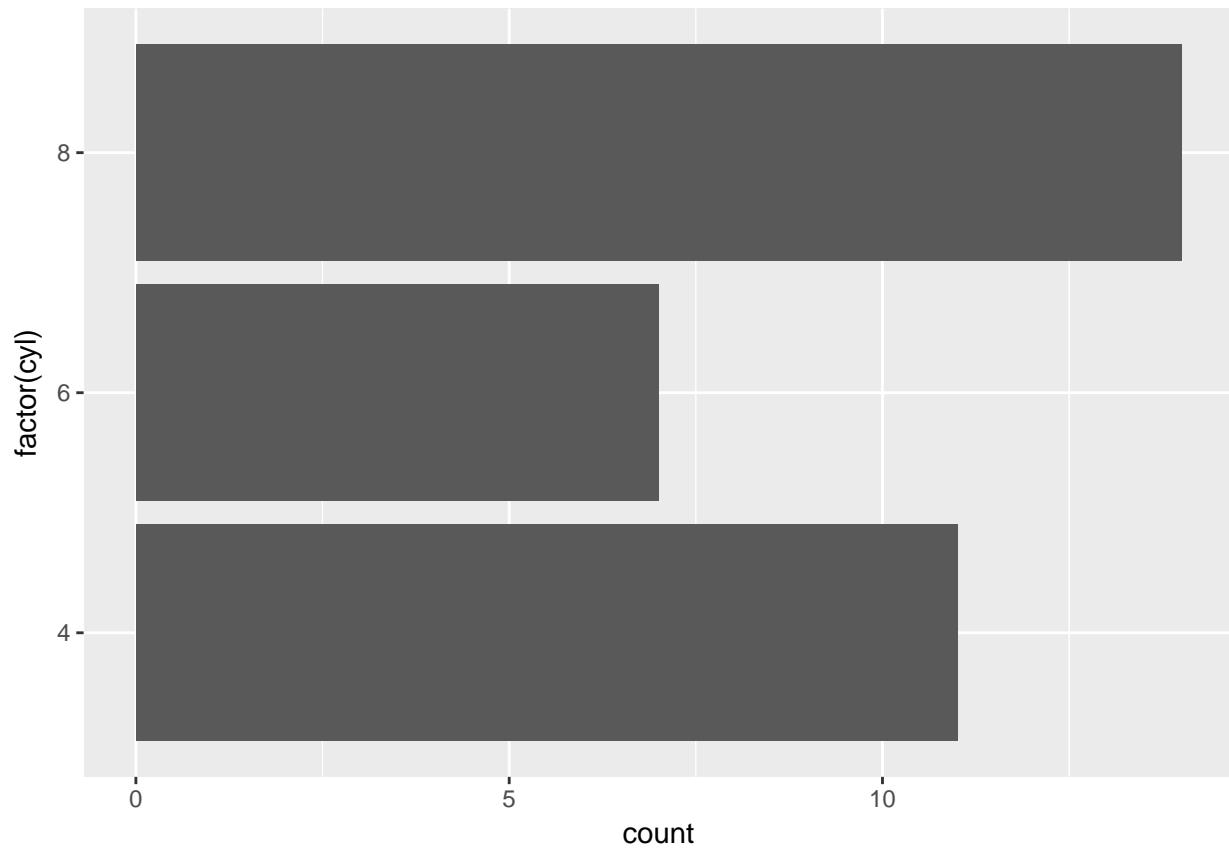
Trendlinie hinzufügen

```
myGG<-qplot(data=diamonds,x=carat,y=depth,color=cut)
myGG + stat_smooth(method="lm")
```



Graphik drehen

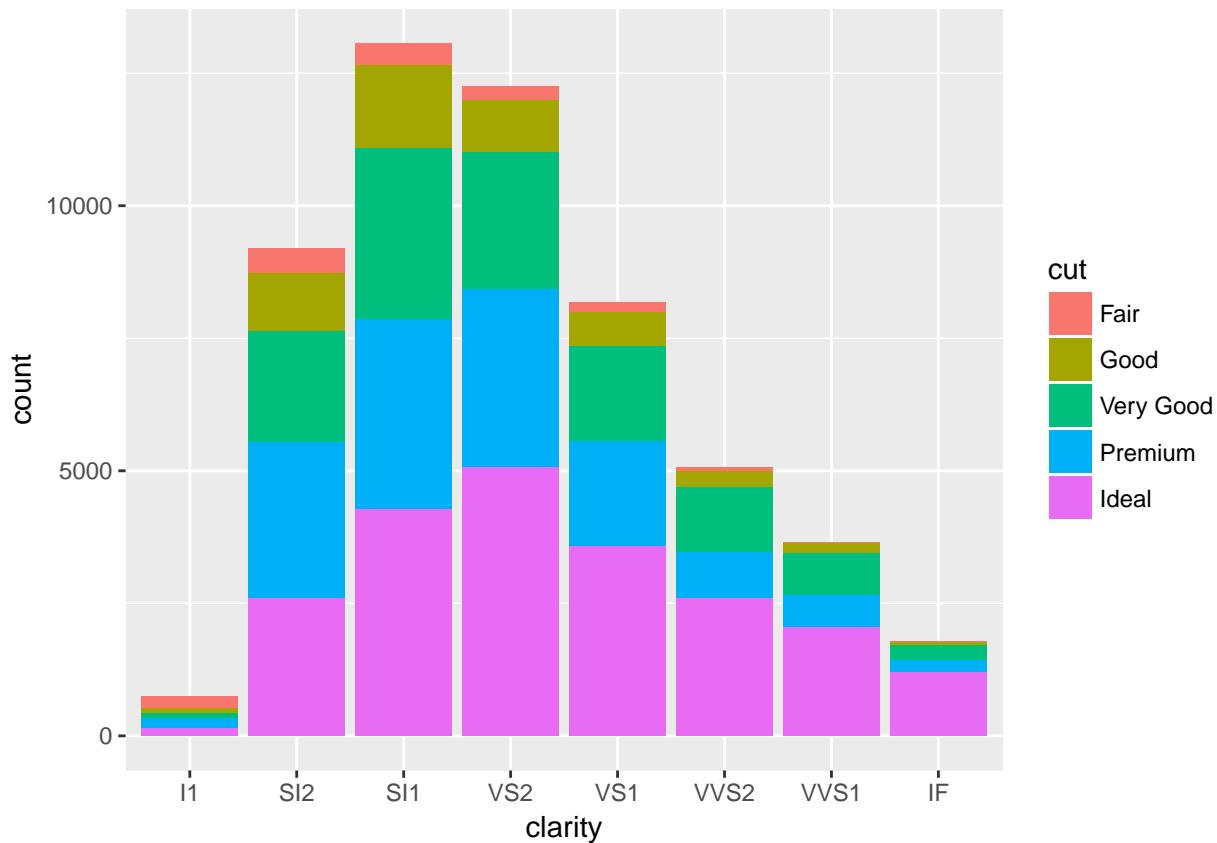
```
qplot(factor(cyl), data=mtcars, geom="bar") +  
coord_flip()
```



Wie nutzt man ggplot

- die aesthetics:

```
ggplot(diamonds, aes(clarity, fill=cut)) + geom_bar()
```



Farben selber wählen

Es wird das Paket `RColorBrewer` verwendet um die Farbpalette zu ändern

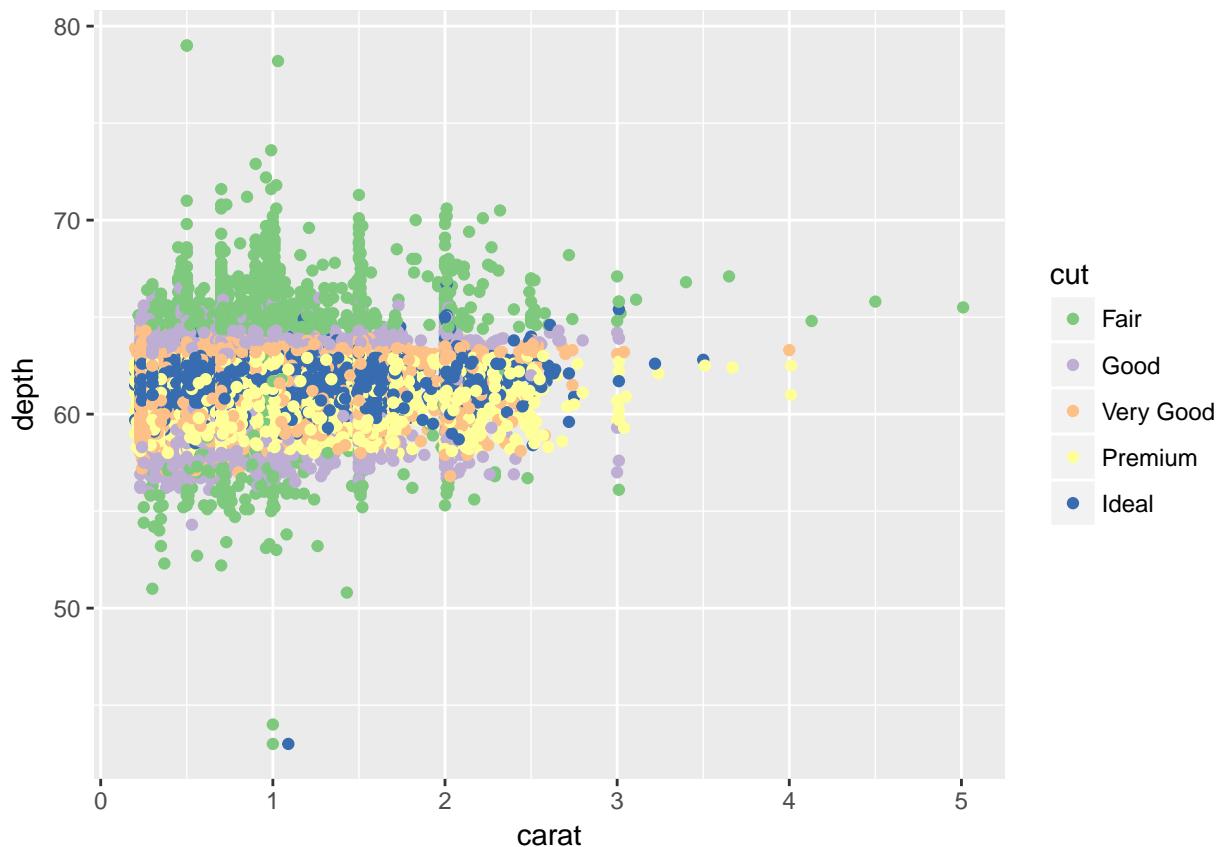
```
install.packages("RColorBrewer")

library(RColorBrewer)
myColors <- brewer.pal(5, "Accent")
names(myColors) <- levels(diamonds$cut)
colScale <- scale_colour_manual(name = "cut",
                                  values = myColors)
```

<http://stackoverflow.com/questions/6919025/>

Eine Graphik mit den gewählten Farben

```
p <- ggplot(diamonds, aes(carat, depth, colour = cut)) +
  geom_point()
p + colScale
```



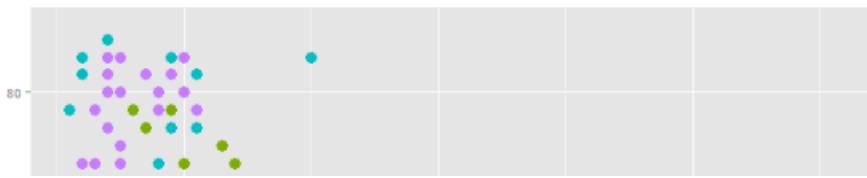
Speichern mit ggsave

```
ggsave("Graphik.jpg")
```

Links

- Warum man ggplot2 für einfache Grafiken nutzen sollte
- Einführung in ggplot2

Introduction to ggplot2



- ggplot2 Basics

- Noam Ross - Quick Introduction to ggplot2
- Plugin ggplot2

Why I use ggplot2

February 12, 2016

By David Robinson



(This article was first published on [Variance Explained](#), and kindly contributed to [R-bloggers](#))



Figure 31:

Einfache Karten mit R erstellen

Gliederung

Arten von räumlichen Daten:

- Straßenkarten
- Satelliten Bilder
- Physische Daten und Karten
- Abstrakte Karten
- ...

Das R-paket ggmap wird im folgenden genutzt um verschiedene Kartentypen darzustellen.

Mit qmap kann man eine schnelle Karte erzeugen.

Straßenkarten

- Straßenkarte werden sehr häufig verwendet.
- Diese Karten zeigen Haupt- und Nebenstraßen (abhängig vom Detail)
- oft sind auch weitere Informationen enthalten. Wie beispielsweise Flughäfen, Städte, Campingplätze oder andere Orte von Interesse
- Beispiel einer Straßenkarte für Mannheim.

Installieren des Paketes

- Zur Erstellung der Karten brauchen wir das Paket ggmap:

```
devtools::install_github("dkahle/ggmap")
devtools::install_github("hadley/ggplot2")
install.packages("ggmap")
```

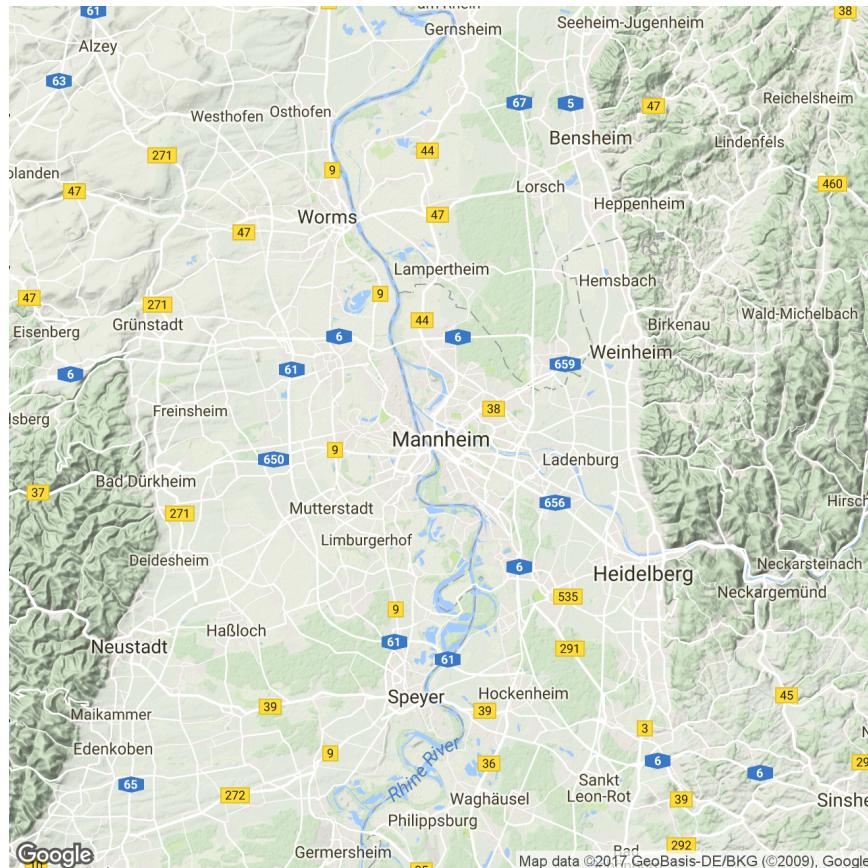
Paket ggmap - Hallo Welt

- Um das Paket zu laden verwenden wir den Befehl `library`

```
library(ggmap)
```

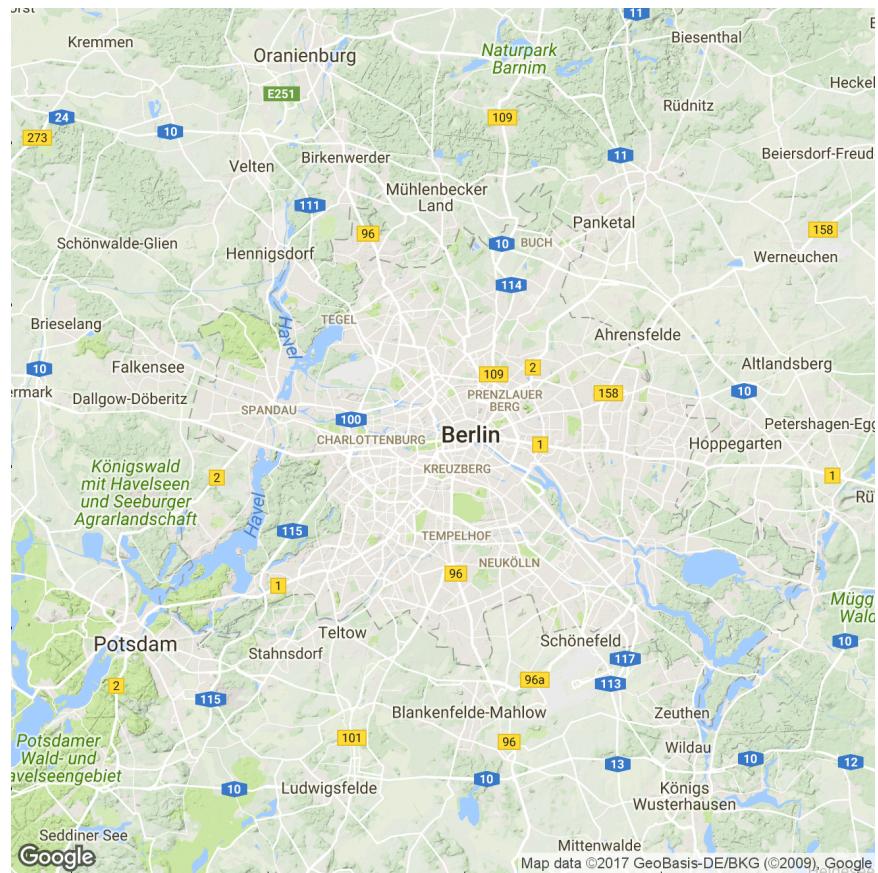
Und schon kann die erste Karte erstellt werden:

```
qmap("Mannheim")
```



Karte für eine Sehenswürdigkeit

```
BBT <- qmap("Berlin Brandenburger Tor")  
BBT
```



Karte für einen ganzen Staat

```
qmap("Germany")
```

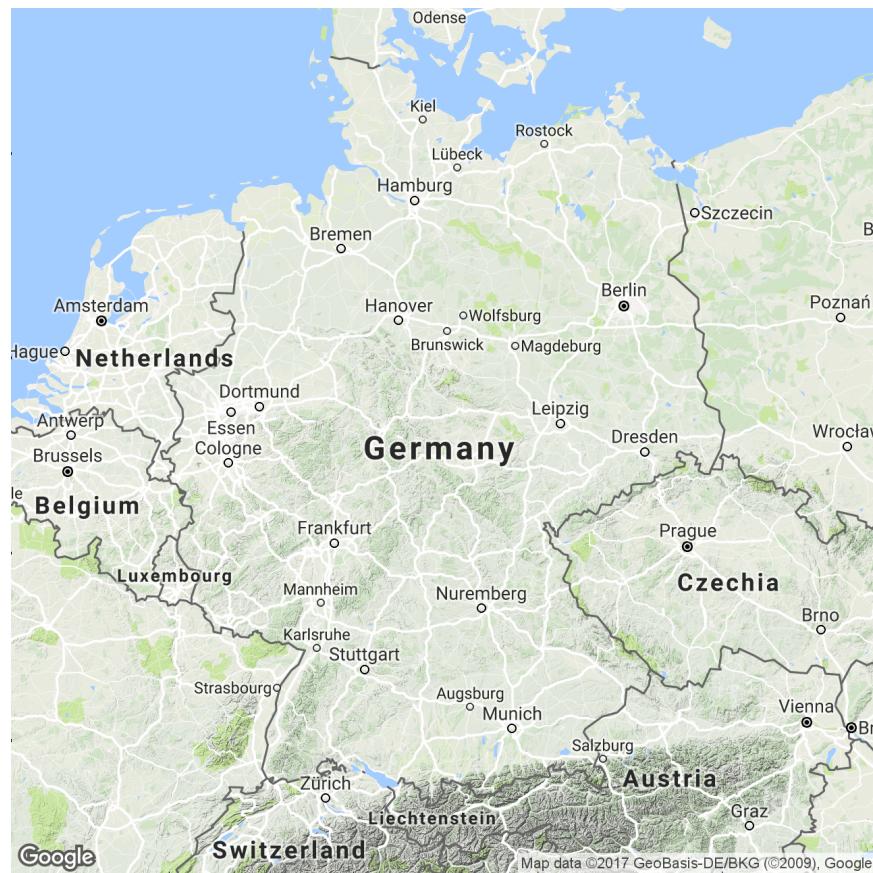


- Wir brauchen ein anderes *zoom level*

Ein anderes *zoom level*

- level 3 - Kontinent
- level 10 - Stadt
- level 21 - Gebäude

```
qmap("Germany", zoom = 6)
```



Hilfe bekommen wir mit dem Fragezeichen

```
?qmap
```

Verschiedene Abschnitte in der Hilfe:

- Description
- Usage
- Arguments
- Value
- Author(s)
- See Also
- Examples

Die Beispiele in der Hilfe

Ausschnitt aus der Hilfe Seite zum Befehl `qmap`:

Das Beispiel kann man direkt in die Konsole kopieren:

```
# qmap("baylor university")
qmap("baylor university", zoom = 14)
# und so weiter
```

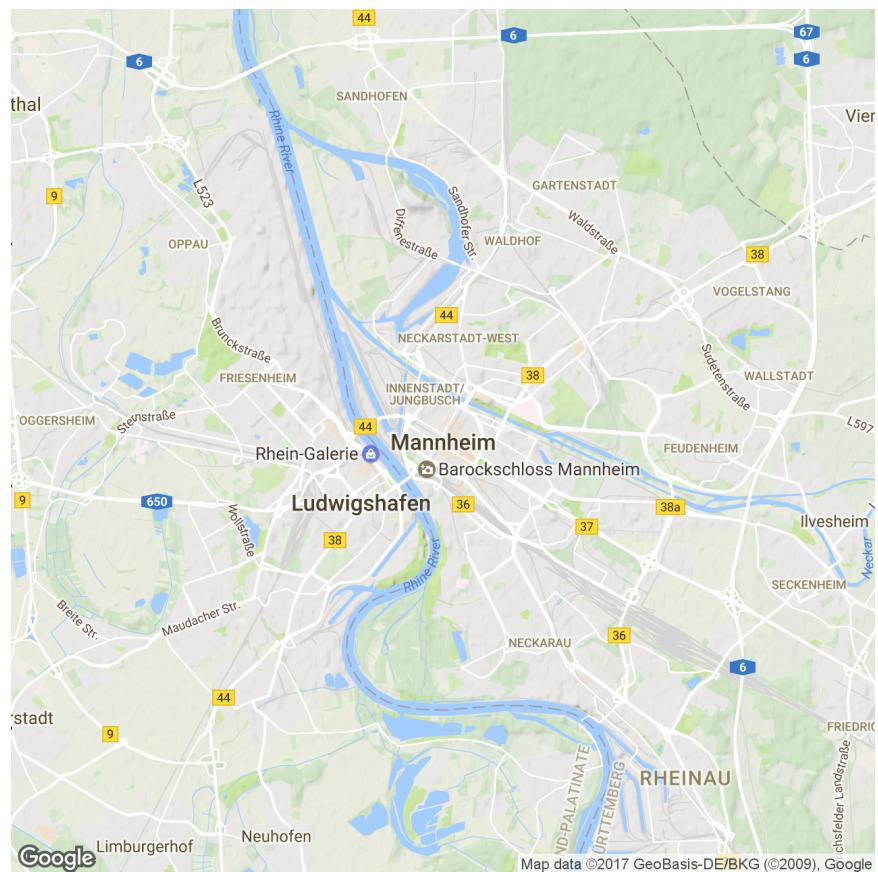
Examples

```
## Not run:  
# these examples have been excluded for checking efficiency  
  
qmap(location = "baylor university")  
qmap(location = "baylor university", zoom = 14)  
qmap(location = "baylor university", zoom = 14, source = "osm")
```

Figure 32: qmap Example

Ein anderes *zoom level*

```
qmap("Mannheim", zoom = 12)
```



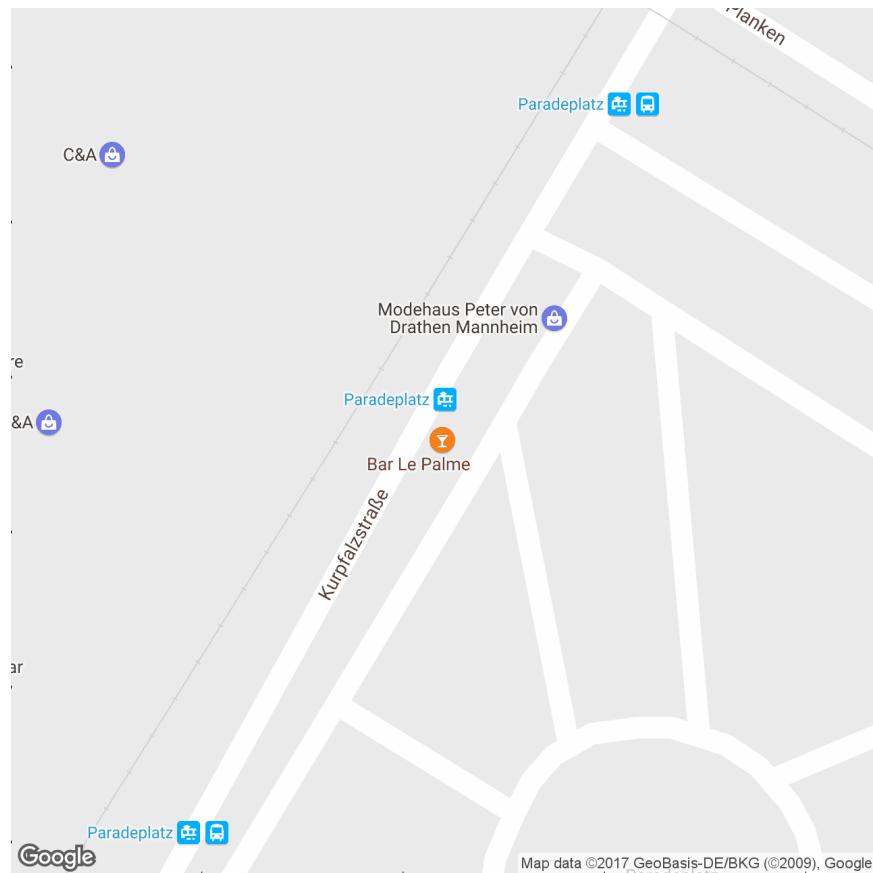
Näher rankommen

```
qmap('Mannheim', zoom = 13)
```



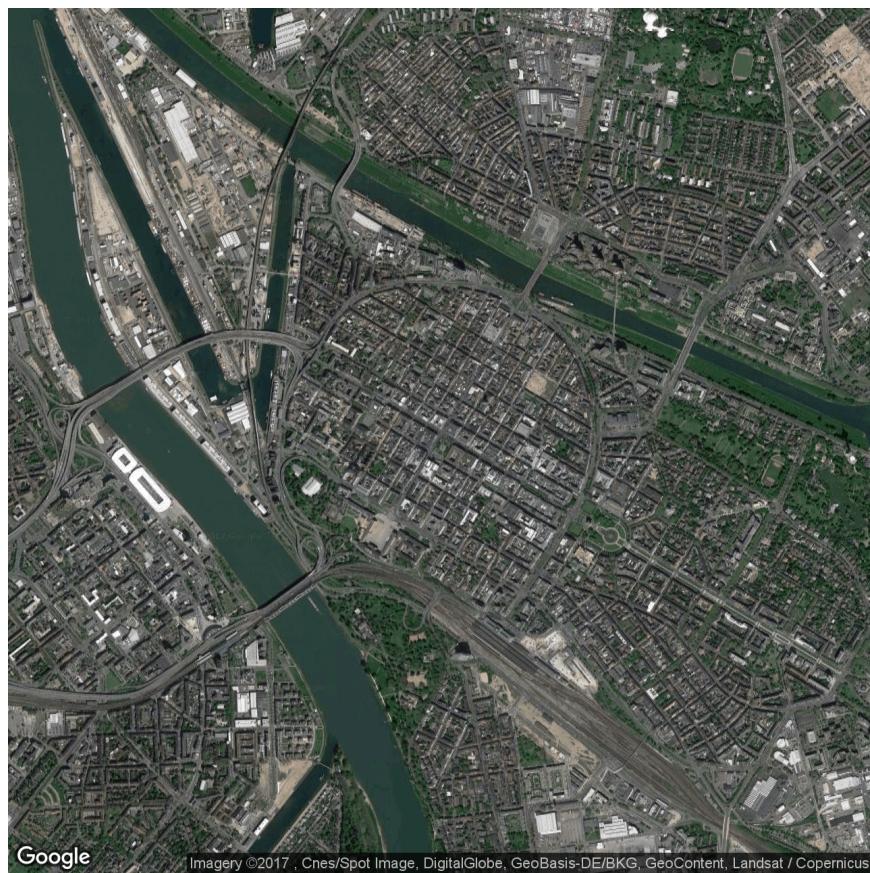
Ganz nah dran

```
qmap('Mannheim', zoom = 20)
```



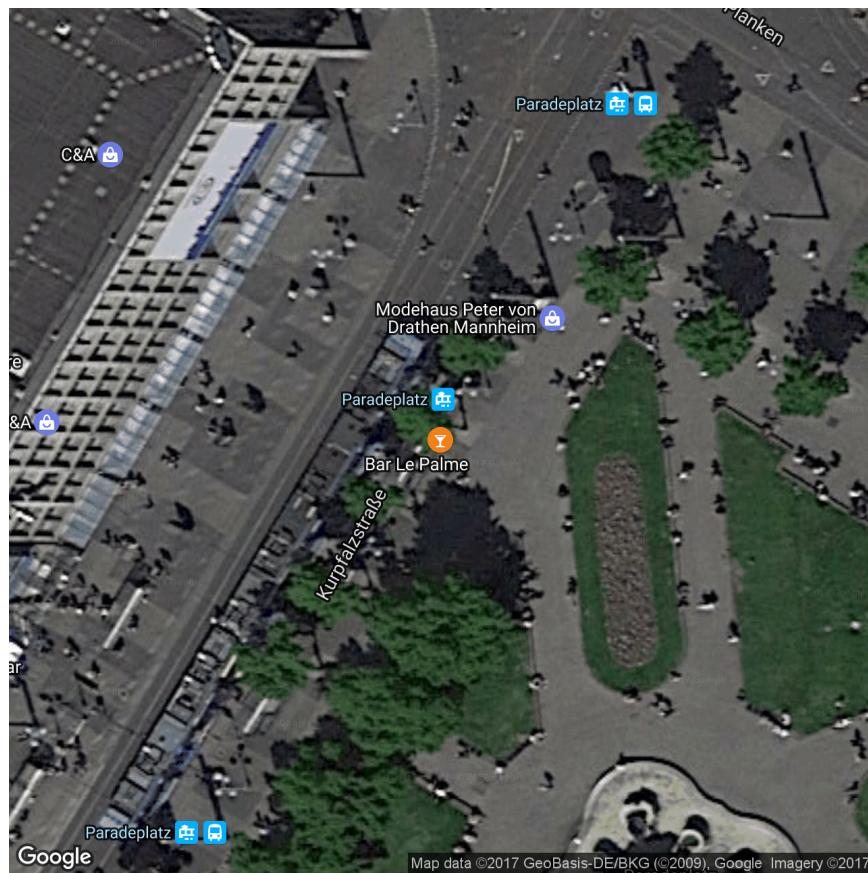
ggmap - maptype satellite

```
qmap('Mannheim', zoom = 14, maptype="satellite")
```



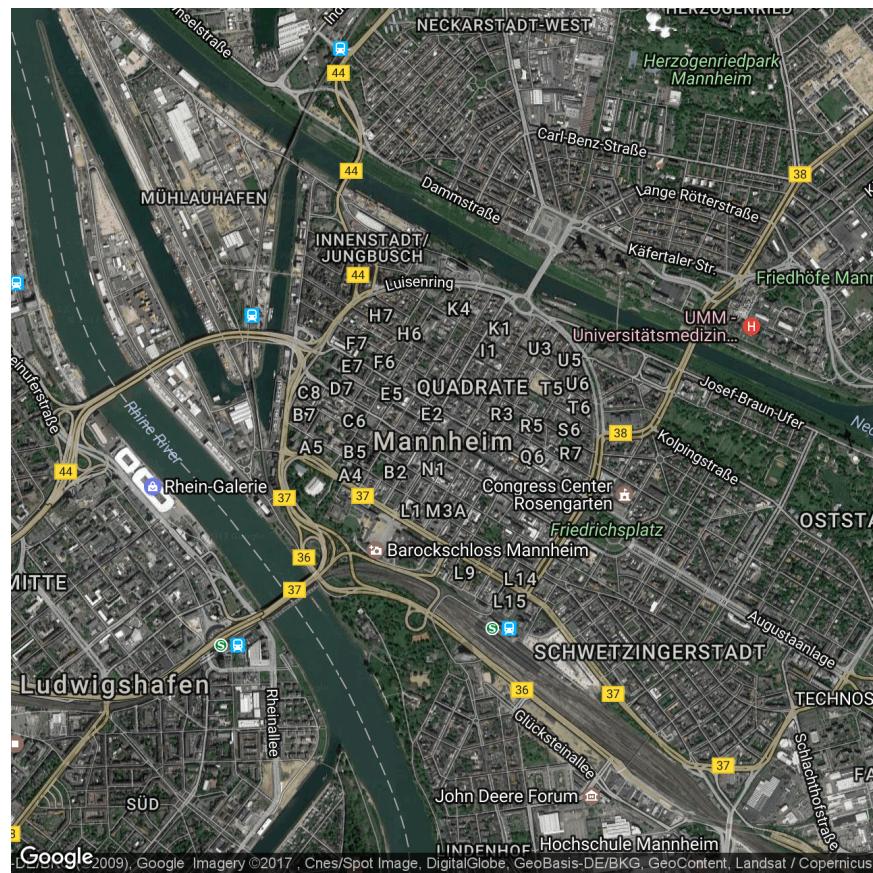
gmap - maptype satellite zoom 20

```
qmap('Mannheim', zoom = 20, maptype="satellite")
```



ggmap - maptype hybrid

```
qmap("Mannheim", zoom = 14, maptype="hybrid")
```

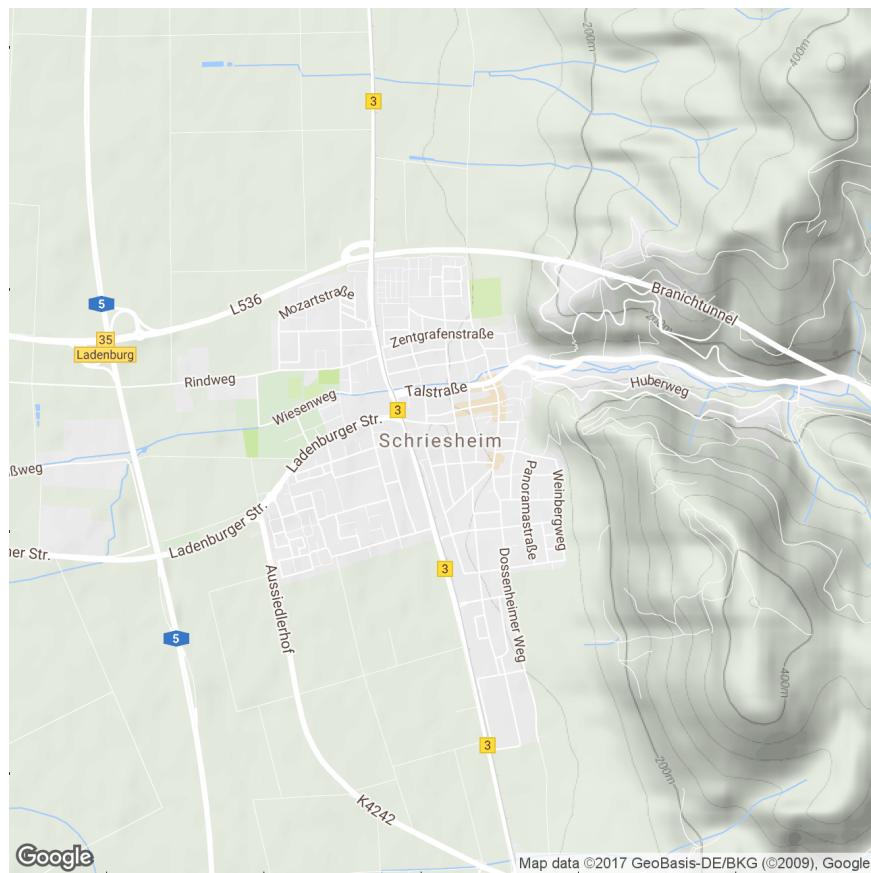


Terrain/physical maps

- Aus Physischen Karten kann man Informationen über Berge, Flüsse und Seen ablesen.
- Farben werden oft genutzt um Höhenunterschiede zu visualisieren

ggmap - terrain map

```
qmap('Schriesheim', zoom = 14, maptype="terrain")
```



Abstrahierte Karten (<http://www.designfaves.com>)



Figure 33: New York

- Abstraktion wird genutzt um nur die essentiellen Informationen einer Karte zu zeigen.
- Bsp. U-Bahn Karten - wichtig sind Richtungen und wenig Infos zur Orientierung

- Im folgenden werden Karten vorgestellt, die sich gut als Hintergrundkarten eignen.

ggmap - maptype watercolor

```
qmap('Mannheim', zoom = 14, maptype="watercolor", source="stamen")
```



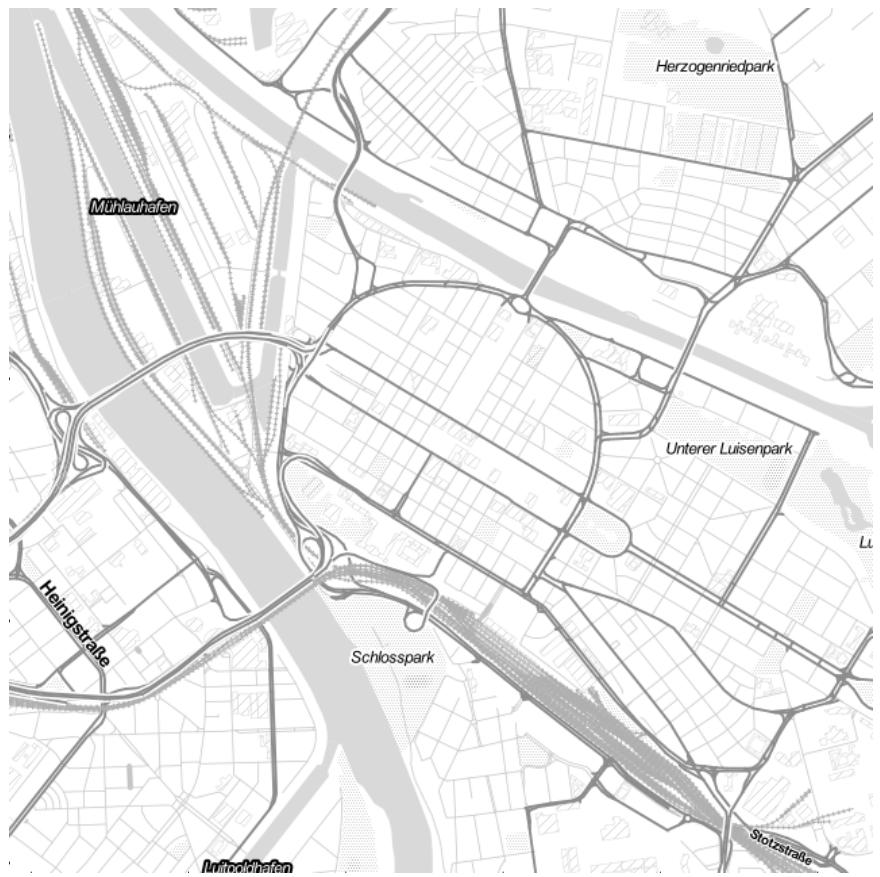
ggmap - source stamen

```
qmap('Mannheim', zoom = 14,  
      maptype="toner", source="stamen")
```



ggmap - maptype toner-lite

```
qmap('Mannheim', zoom = 14,  
      maptype="toner-lite", source="stamen")
```



ggmap - maptype toner-hybrid

```
qmap('Mannheim', zoom = 14,  
      maptype="toner-hybrid", source="stamen")
```



ggmap - maptype terrain-lines

```
qmap('Mannheim', zoom = 14,  
      maptype="terrain-lines",source="stamen")
```



Graphiken speichern

ggmap - ein Objekt erzeugen

- <- ist der Zuweisungspfeil um ein Objekt zu erzeugen
- Dieses Vorgehen macht bspw. Sinn, wenn mehrere Karten nebeneinander gebraucht werden.

```
MA_map <- qmap('Mannheim',
                  zoom = 14,
                  maptype="toner",
                  source="stamen")
```

Geokodierung

Geocoding (...) uses a description of a location, most typically a postal address or place name, to find geographic coordinates from spatial reference data ...

Wikipedia - Geocoding

```
library(ggmap)
geocode("Mannheim",source="google")
```

	lon	lat
	8.463243	49.48604

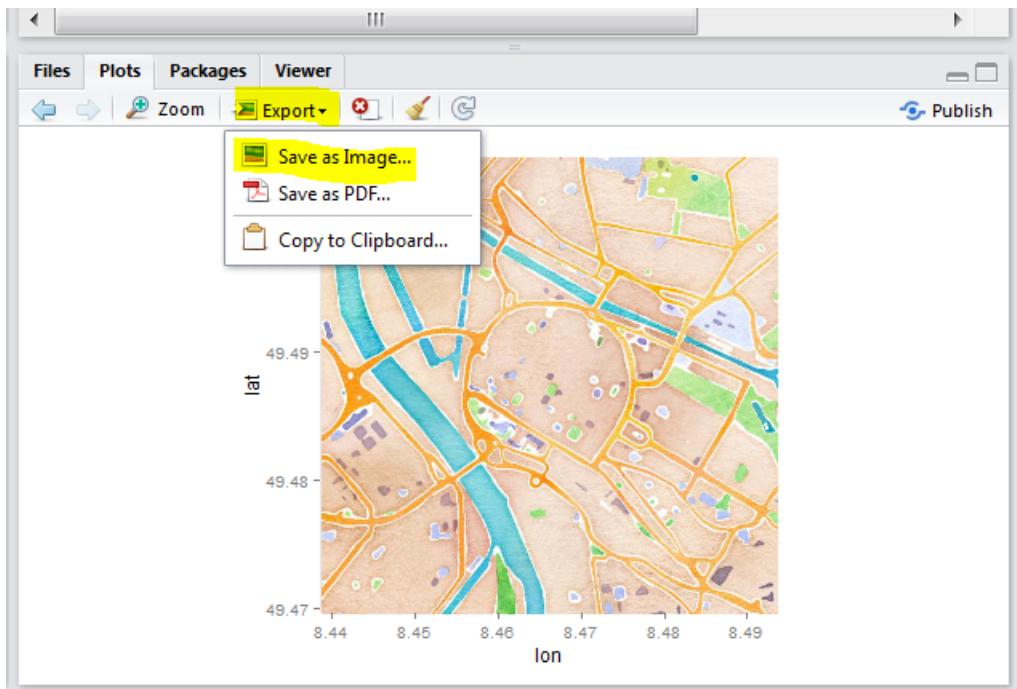


Figure 34: RstudioExport

Latitude und Longitude

<http://modernsurvivalblog.com>

Koordinaten verschiedener Orte in Deutschland

cities	lon	lat
Hamburg	9.993682	53.55108
Koeln	6.960279	50.93753
Dresden	13.737262	51.05041
Muenchen	11.581981	48.13513

Reverse Geokodierung

Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighbourhoods, county, state, or country.

Quelle: Wikipedia

```
revgeocode(c(48,8))
```

```
## [1] "Unnamed Road, Somalia"
```

Die Distanz zwischen zwei Punkten

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim")
```

```
##          from      to    m    km 10 miles seconds minutes
## 1 Q1, 4 Mannheim B2, 1 Mannheim 749 0.749 0.4654286     212 3.533333
##          hours
## 1 0.05888889
```

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim", mode="walking")
```

LATITUDE

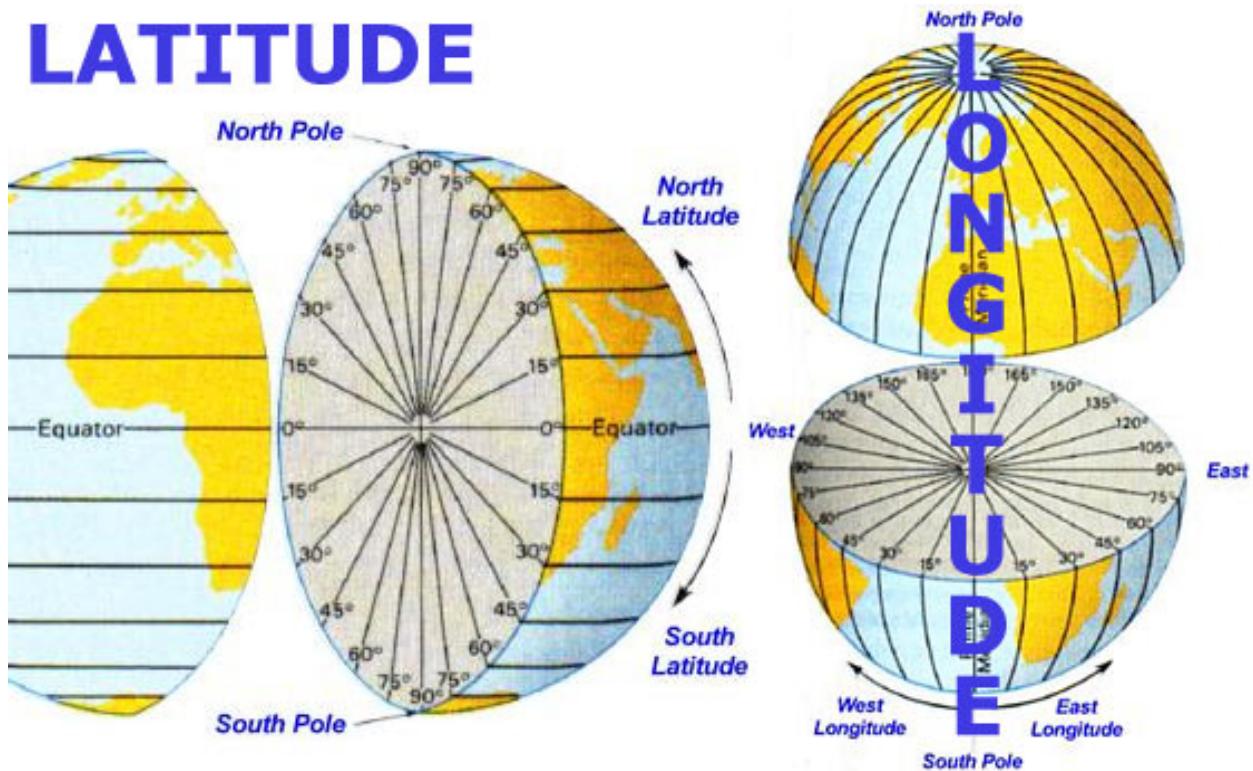


Figure 35: LatLon

Geokodierung - verschiedene Punkte von Interesse

```
POI1 <- geocode("B2, 1 Mannheim",source="google")
POI2 <- geocode("Hbf Mannheim",source="google")
POI3 <- geocode("Mannheim, Friedrichsplatz",source="google")
ListPOI <-rbind(POI1,POI2,POI3)
POI1;POI2;POI3

##           lon      lat
## 1 8.462844 49.48569
##           lon      lat
## 1 8.469879 49.47972
##           lon      lat
## 1 8.475208 49.48326
```

Punkte in der Karte

```
MA_map +
geom_point(aes(x = lon, y = lat),
data = ListPOI)
```



Punkte in der Karte

```
MA_map +  
geom_point(aes(x = lon, y = lat), col="red",  
data = ListPOI)
```



ggmap - verschiedene Farben

```
ListPOI$color <- c("A", "B", "C")
MA_map +
  geom_point(aes(x = lon, y = lat, col = color),
  data = ListPOI)
```



ggmap - größere Punkte

```
ListPOI$size <- c(10,20,30)
MA_map +
  geom_point(aes(x = lon, y = lat, col=color, size=size),
  data = ListPOI)
```



Eine Route von Google maps bekommen

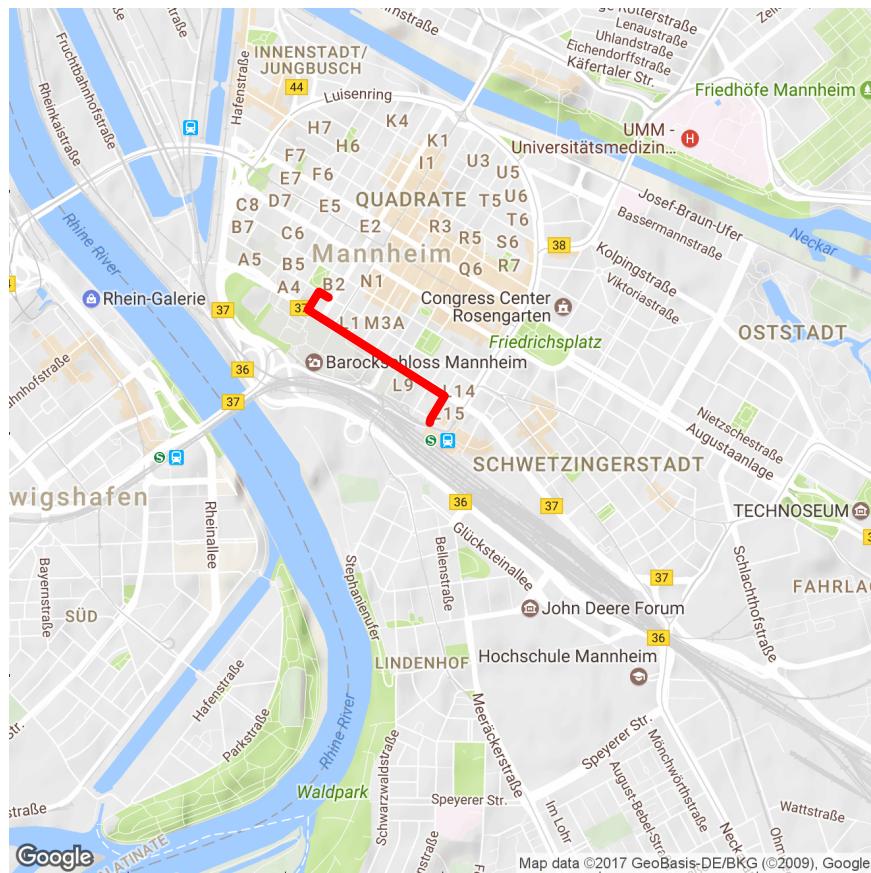
```
from <- "Mannheim Hbf"
to <- "Mannheim B2 , 1"
route_df <- route(from, to, structure = "route")
```

Mehr Information

<http://rpackages.ianhowson.com/cran/ggmap/man/route.html>

Eine Karte mit dieser Information zeichnen

```
qmap("Mannheim Hbf", zoom = 14) +
  geom_path(
    aes(x = lon, y = lat), colour = "red", size = 1.5,
    data = route_df, lineend = "round"
  )
```



Wie fügt man Punkte hinzu

- Nutzung von geom_point
- Question on stackoverflow

<http://i.stack.imgur.com>

Cheatsheet

- Cheatsheet zu data visualisation

<https://www.rstudio.com/>

Resourcen und Literatur

- Artikel von David Kahle und Hadley Wickham zur Nutzung von ggmap.
- Schnell eine Karte bekommen
- Karten machen mit R
- Problem mit der Installation von ggmap

Take Home Message

Was klar sein sollte:

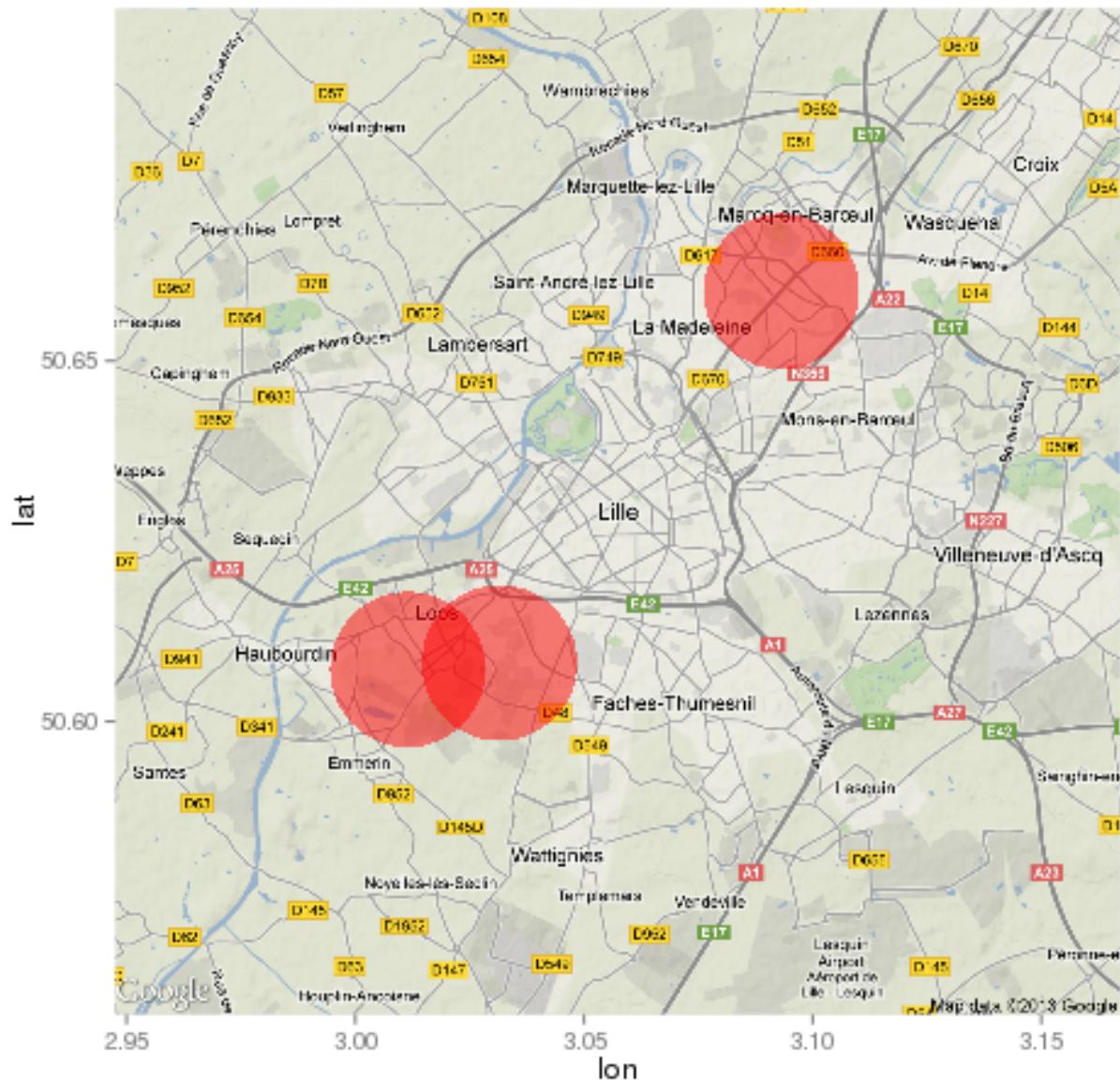


Figure 36: pic

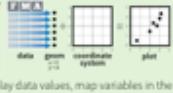
Data Visualization with ggplot2

Cheat Sheet

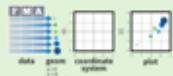


Basics

ggplot2 is based on the [grammar of graphics](#), the idea you can build every graph from the same few components: a **data** set, a set of **geoms**—visual objects that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

```
qplot(x + cty, y + hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.
```

```
ggplot(data = mpg, aes(x = cty, y = hwy))
```

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
geom_<type>(<parameters>)
Add layers, elements with <type>, and position adjustment.
```

```
last_plot()
Returns the last plot
```

```
ggsave("plot.png", width = 5, height = 5)
```

Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

```
a <- ggplot(mpg, aes(hwy))
```

```
#+ geom_area(stat = "bin")  
x, y, alpha, color, fill, linetype, size  
b + geom_area(stat = "density")  
x, y, alpha, color, fill, linetype, size, weight  
b + geom_density( kernel = "gaussian")  
x, y, alpha, color, fill, linetype, size, weight  
b + geom_density_2d(..., county = 1)
```

```
#+ geom_dotplot()
```

```
x, y, alpha, color, fill
```

```
#+ geom_freqpoly()
```

```
x, y, alpha, color, linetype, size  
b + geom_freqpoly(aes(x = density...))
```

```
#+ geom_histogram(binwidth = 5)
```

```
x, y, alpha, color, fill, linetype, size, weight  
b + geom_histogram(aes(density = ...))
```

Discrete

```
b <- ggplot(mpg, aes(cty))
```

```
#+ geom_bar()
```

```
x, y, alpha, color, fill, linetype, size, weight
```

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
```

```
#+ geom_blank()
```

```
#+ geom_jitter()
```

```
x, y, alpha, color, fill, shape, size
```

```
#+ geom_point()
```

```
x, y, alpha, color, fill, shape, size
```

```
#+ geom_quantile()
```

```
x, y, alpha, color, linetype, size, weight
```

```
#+ geom_rug(sides = "lf")
```

```
alpha, color, linetype, size
```

```
#+ geom_smooth(method = lm)
```

```
x, y, alpha, color, fill, linetype, size, weight
```

```
#+ geom_smooth(method = "loess")
```

```
x, y, alpha, color, fill, linetype, size, weight
```

```
#+ geom_text(aes(label = cty))
```

```
x, y, label, alpha, angle, color, family, fontface,
```

```
height, lineheight, size, vjust
```

```
A B
```

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
```

```
#+ geom_bin2d(binwidth = c(5, 0.5))
```

```
xmax, xmin, ymax, ymin, alpha, color, fill,
```

```
linetype, size, weight
```

```
#+ geom_density2d()
```

```
x, y, alpha, colour, linetype, size
```

```
#+ geom_hex()
```

```
x, y, alpha, colour, fill, size
```

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
```

```
#+ geom_area()
```

```
x, y, alpha, color, fill, linetype, size
```

```
#+ geom_line()
```

```
x, y, alpha, color, linetype, size
```

```
#+ geom_step(direction = "hv")
```

```
x, y, alpha, color, linetype, size
```

Visualizing error

```
df <- data.frame(gpp = c("A", "B"), fit = 4.5, se = 1.2)
```

```
k <- ggplot(df, aes(gpp, fit, ymin = fit - se, ymax = fit + se))
```

```
#+ geom_crossbar(fatten = 2)
```

```
x, y, ymax, ymin, alpha, color, fill, linetype,
```

```
#+ geom_errorbar()
```

```
x, ymin, ymax, alpha, color, linetype, size,
```

```
width (also geom_errorbarh())
```

```
#+ geom_linerange()
```

```
x, ymin, ymax, alpha, color, linetype, size
```

```
#+ geom_pointrange()
```

```
x, y, ymin, ymax, alpha, color, fill, linetype,
```

```
shape, size, weight
```

Maps

```
data <- data.frame(murder = USArrests$Murder,
```

```
state = USArrests$State, arrests = USArrests$Arrests[])
```

```
murder_mean <- mean(murder)
```

```
1 <- ggplot(data, aes(id = murder))
```

```
#+ geom_map(aes(id = map), map = map) +
```

```
expand_limits(x = map$long, y = map$lat)
```

```
map$id, alpha, color, fill, linetype, size
```

Three Variables

```
sealsSz <- width(seals, sqrt(delta_long^2 + delta_lat^2))
```

```
m <- ggplot(seals, aes(lat, long))
```

```
#+ geom_rect(aes(xmin = long, ymin = lat,
```

```
xmax = long + delta_long,
```

```
ymax = lat + delta_lat)
```

```
xmax, xmin, ymax, ymin, alpha, color, fill,
```

```
linetype, size
```

```
#+ geom_raster(aes(z = z), hjust = 0.5,
```

```
vjust = 0.5, interpolate = FALSE)
```

```
x, y, alpha, fill
```

```
#+ geom_contour(aes(z = z))
```

```
x, y, alpha, colour, linetype, size, weight
```

```
#+ geom_tile(aes(z = z))
```

```
x, y, alpha, color, fill, linetype, size
```

```
Learn more at: docs.ggplot2.org • ggplot2 0.9.3.1 • Updated: 3/15
```

Figure 37: Cheatsheet

- Wie man eine schnelle Karte erzeugt
- Wie man geokodiert
- Wie man eine Distanz berechnet

Regressionsdiagnostik mit R-Paket `visreg`

Regressionsdiagnostik mit Basis-R

Ein einfaches Modell

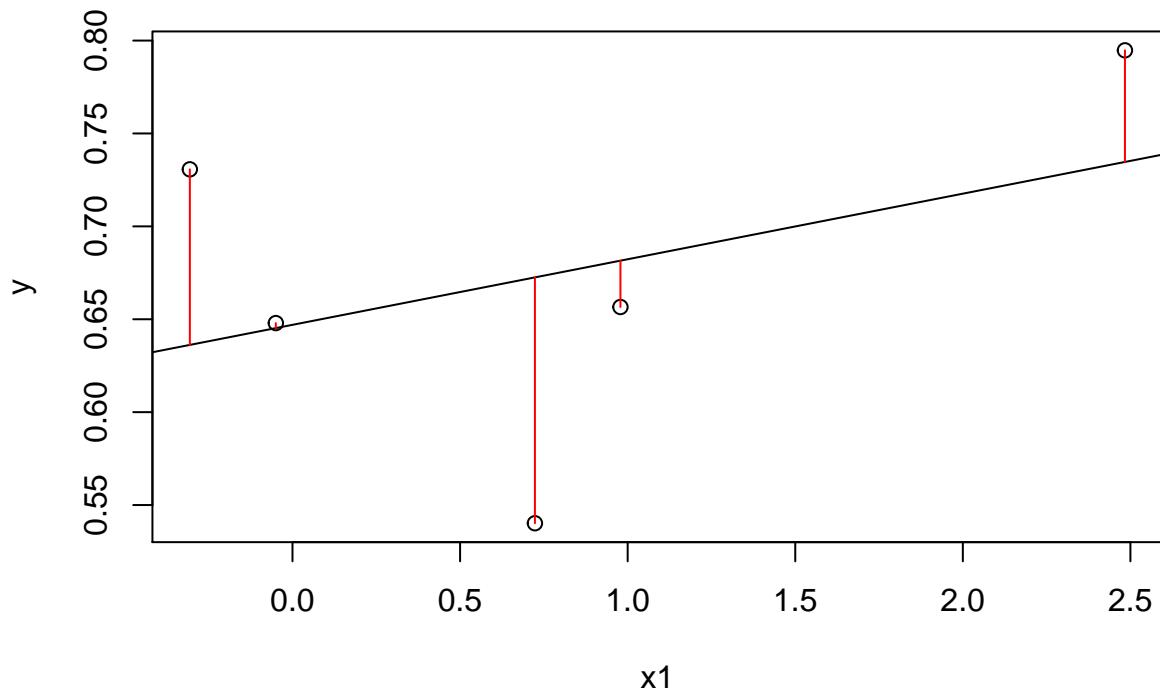
```
N <- 5
x1 <- rnorm(N)
y <- runif(N)
```

Modellvorhersage machen

```
mod1 <- lm(y~x1)
pre <- predict(mod1)
```

Regressionsdiagnostik mit Basis-R

```
plot(x1,y)
abline(mod1)
segments(x1, y, x1, pre, col="red")
```



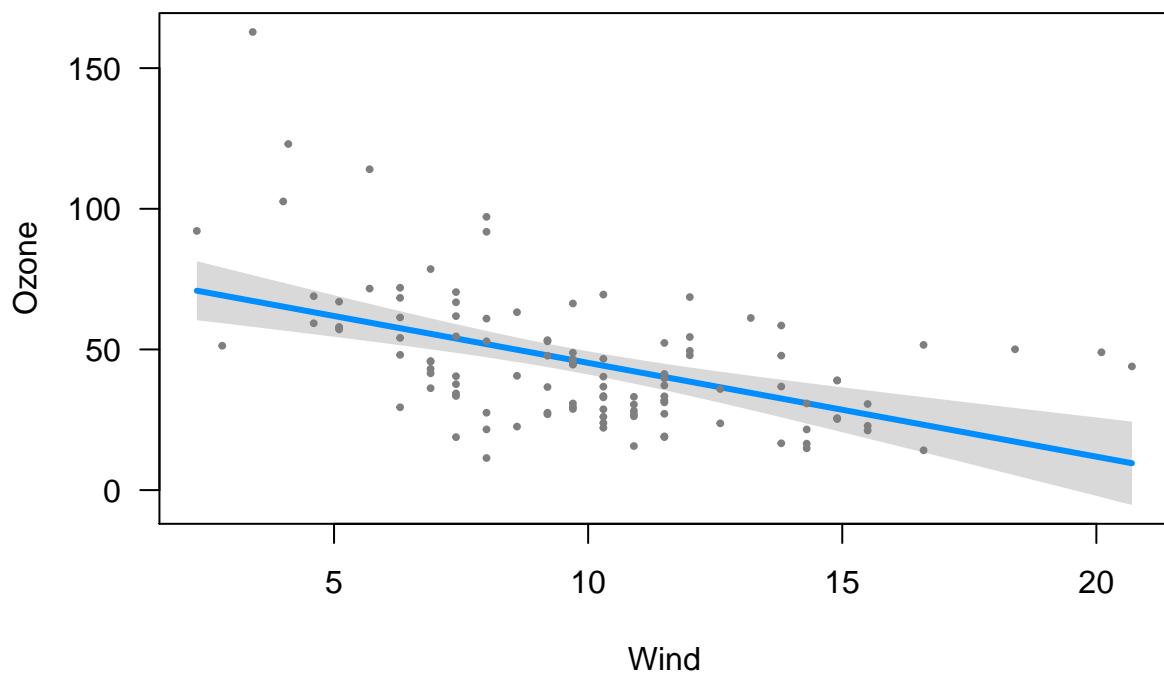
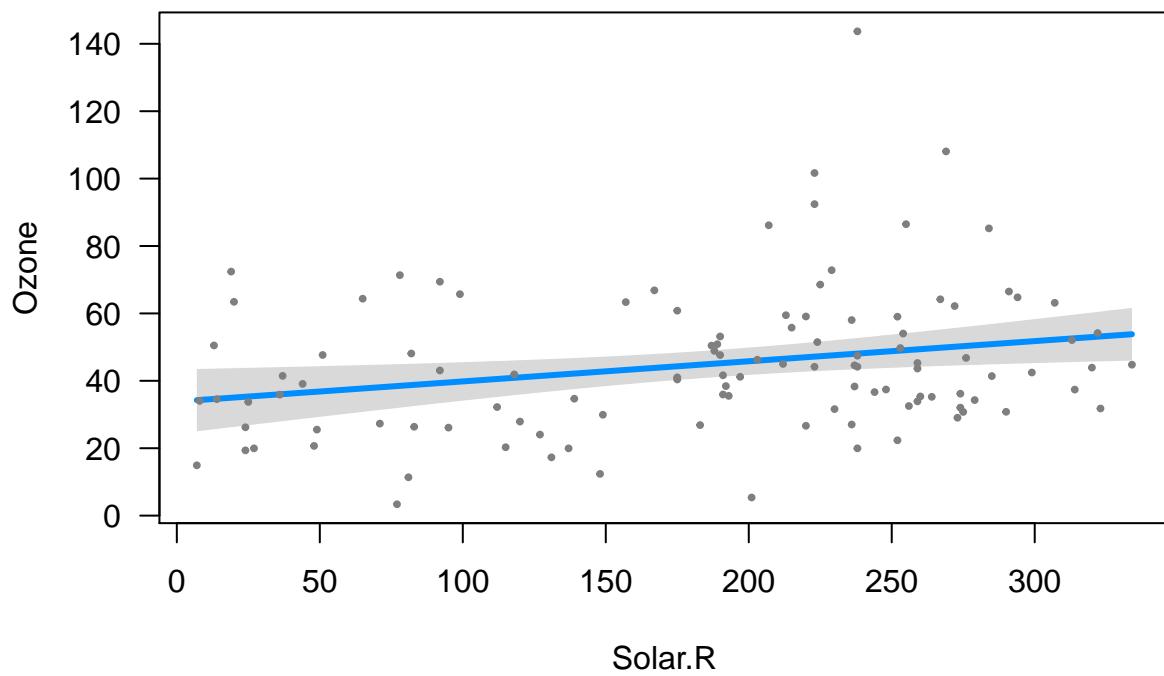
Das visreg-Paket

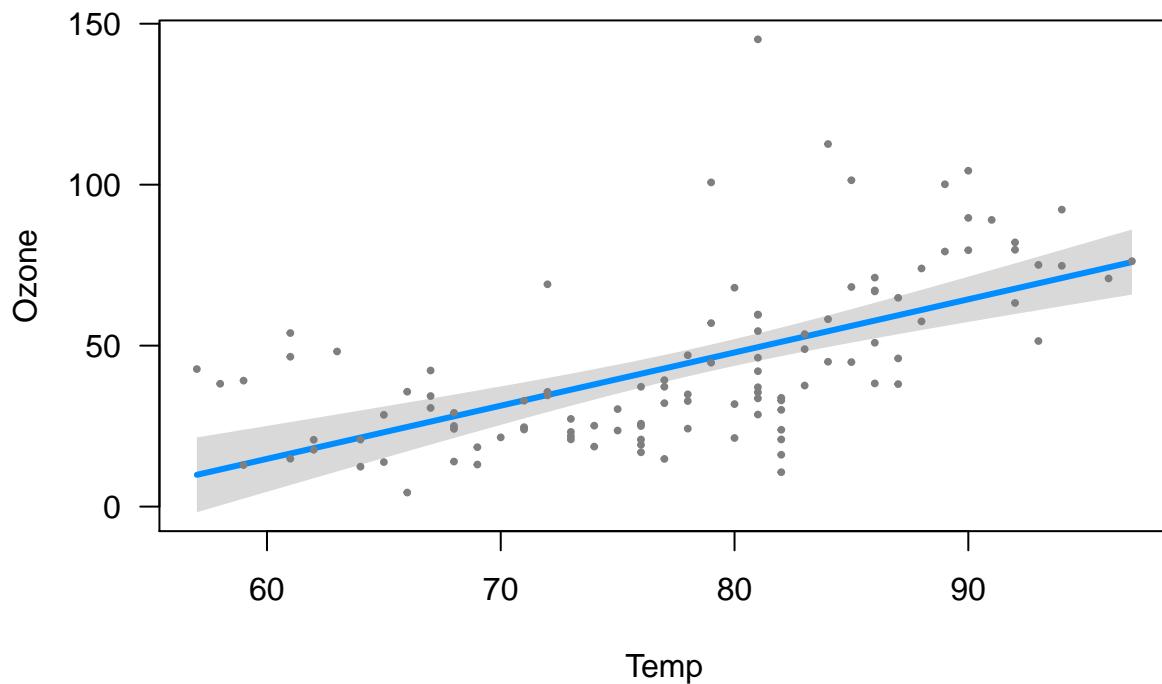
Ein Modell wird auf dem airquality Datensatz geschätzt

```
install.packages("visreg")  
  
library(visreg)  
fit <- lm(Ozone ~ Solar.R + Wind + Temp, data = airquality)
```

Visualisierung

```
visreg(fit)
```





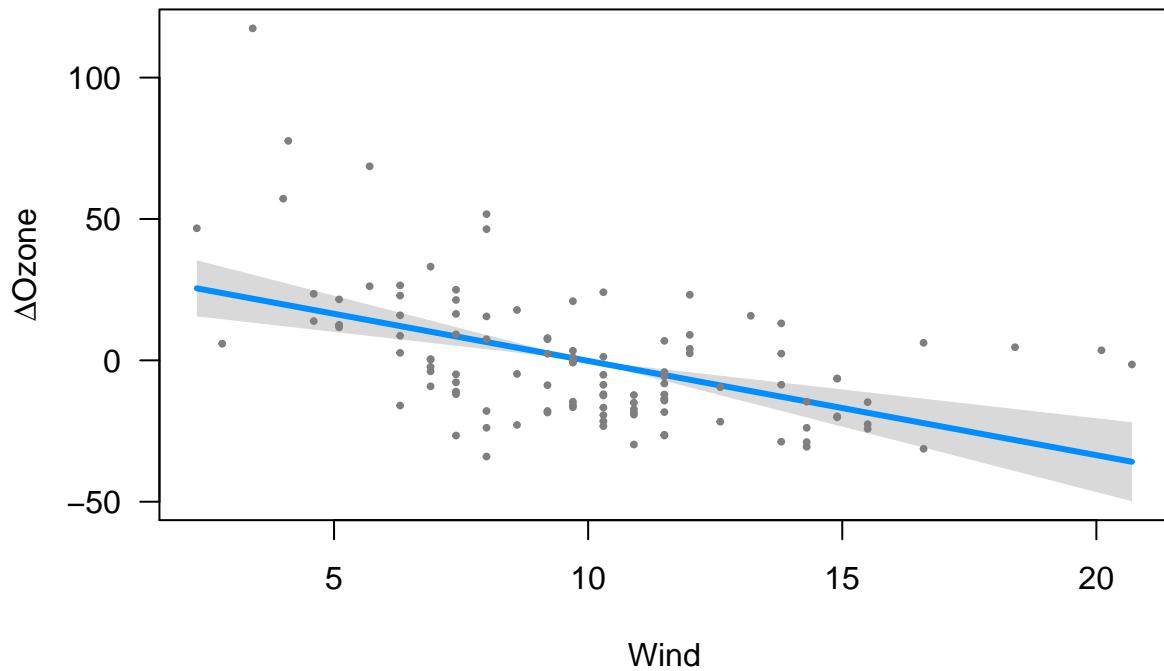
Und dann mit `visreg` visualisiert.

- Zweites Argument - Spezifikation erklärende Variable für Visualisierung

```
visreg(fit, "Wind", type = "contrast")
```

Visualisierung mit dem Paket `visreg`

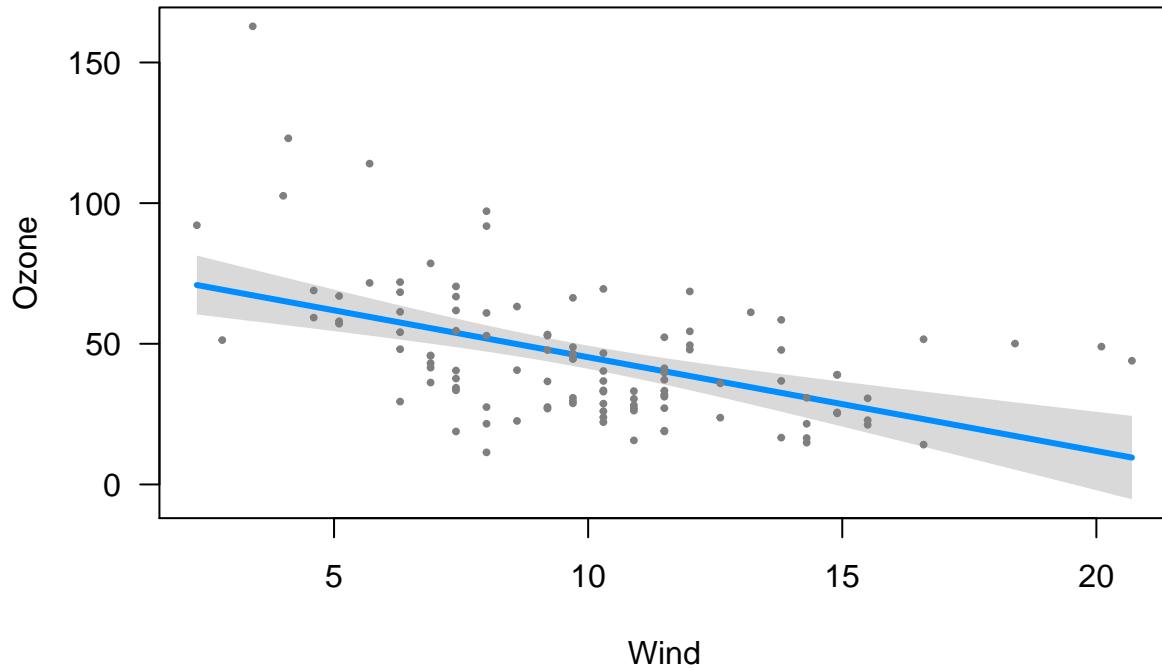
```
visreg(fit, "Wind", type = "contrast")
```



Das visreg-Paket

- Das Default-Argument für type ist conditional.

```
visreg(fit, "Wind", type = "conditional")
```



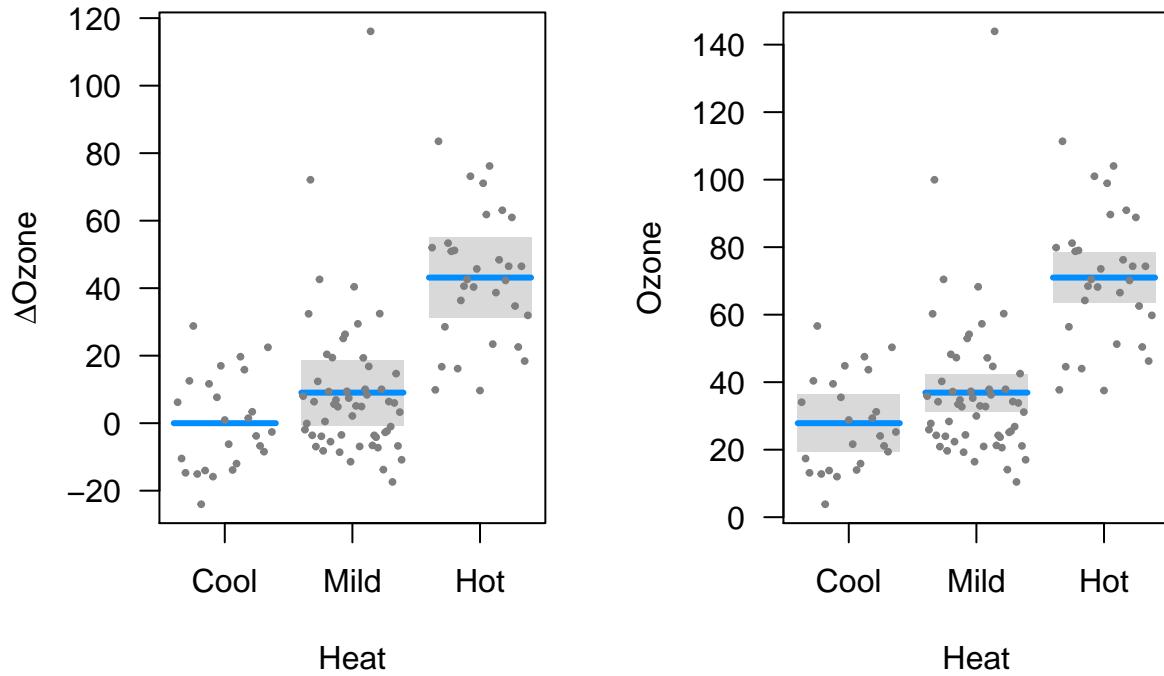
Regression mit Faktoren

Mit visreg können die Effekte bei Faktoren visualisiert werden.

```
airquality$Heat <- cut(airquality$Temp, 3,
  labels=c("Cool", "Mild", "Hot"))
fit.heat <- lm(Ozone ~ Solar.R + Wind + Heat,
  data = airquality)
```

Effekte von Faktoren

```
par(mfrow=c(1,2))
visreg(fit.heat, "Heat", type = "contrast")
visreg(fit.heat, "Heat", type = "conditional")
```

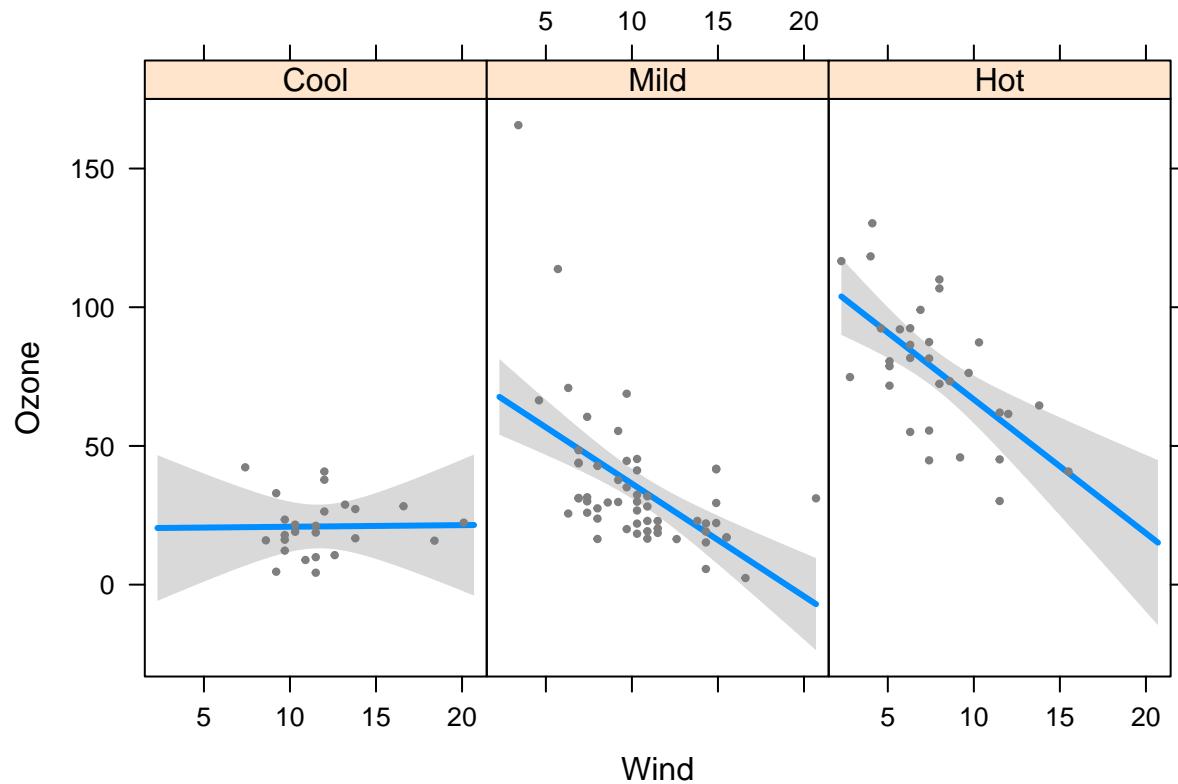


Das Paket visreg - Interaktionen

```
airquality$Heat <- cut(airquality$Temp, 3,
labels=c("Cool", "Mild", "Hot"))
fit <- lm(Ozone ~ Solar.R + Wind * Heat, data = airquality)
```

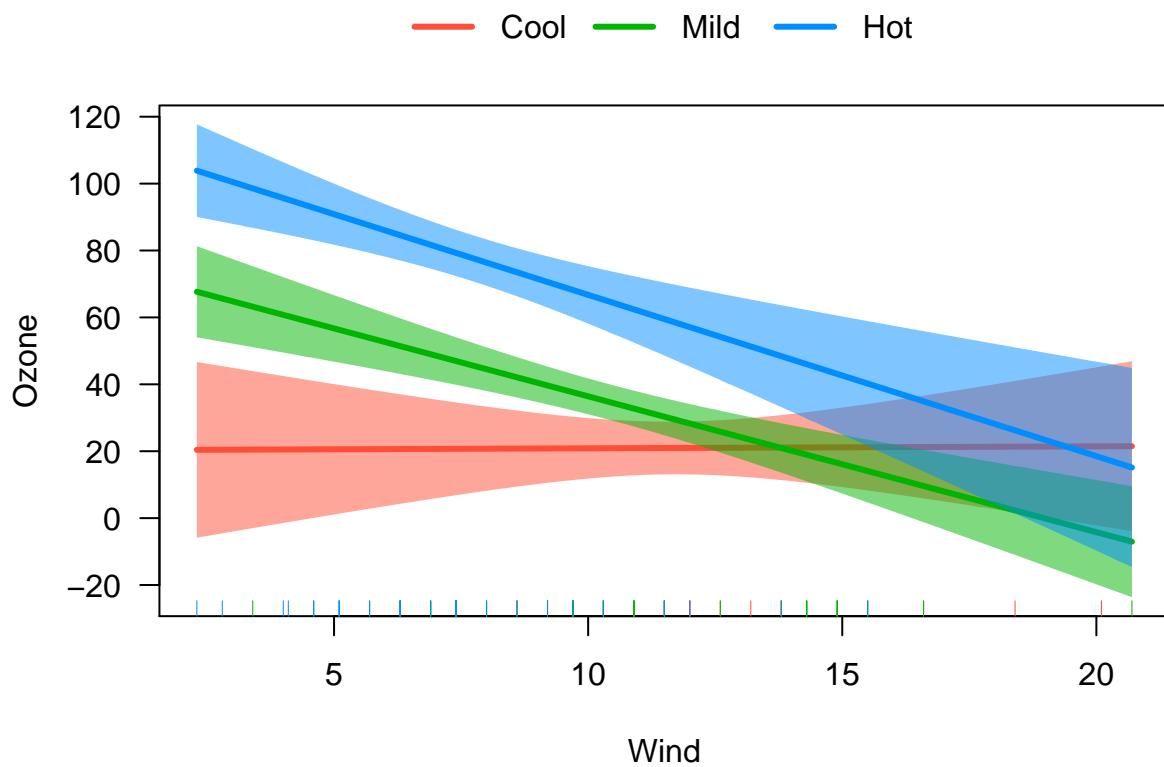
Steuern der Graphikausgabe mittels layout

```
visreg(fit, "Wind", by = "Heat", layout=c(3,1))
```



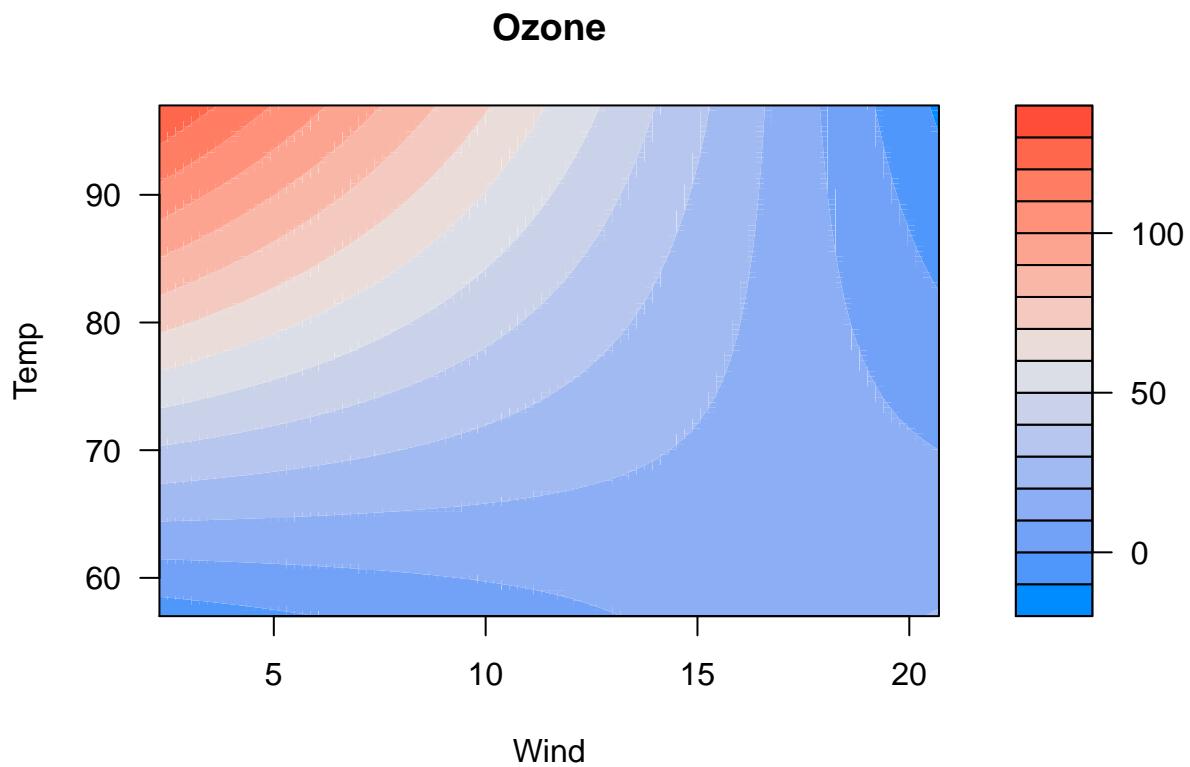
Das Paket visreg - Interaktionen overlay

```
fit <- lm(Ozone ~ Solar.R + Wind * Heat, data = airquality)
visreg(fit, "Wind", by="Heat", overlay=TRUE, partial=FALSE)
```



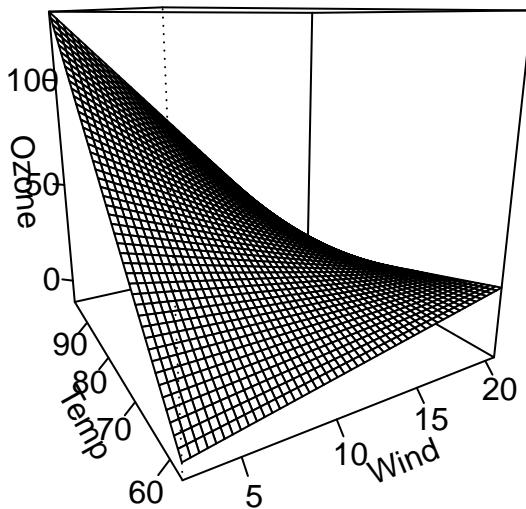
Das Paket visreg - visreg2d

```
fit2 <- lm(Ozone ~ Solar.R + Wind * Temp, data = airquality)
visreg2d(fit2, "Wind", "Temp", plot.type = "image")
```



Das Paket visreg - surface

```
visreg2d(fit2, "Wind", "Temp", plot.type = "persp")
```



Weitere Themen im Ausblick

Mehr User Interface - Der R-commander

```
library("Rcmdr")
```

Interaktive Grafiken mit R

Das Paket ipplots

```
install.packages("ipplots", dep=TRUE)
```

- Das Paket laden:

```
library(ipplots)
```

```
cyl.f <- factor(mtcars$cyl)
gear.f <- factor(mtcars$factor)
attach(mtcars)
ihist(mpg) # histogram
ibar(carb) # barchart
iplot(mpg, wt) # scatter plot
ibox(mtcars[c("qsec", "disp", "hp")]) # boxplots
```

```
ipcp(mtcars[c("mpg","wt","hp")]) # parallel coordinates  
imosaic(cyl.f,gear.f) # mosaic plot
```

R-Paket rggobi

```
library(rggobi)  
g <- ggobi(mydata)
```

Interaktion mit plots

```
attach(mydata)  
plot(x, y) # scatterplot  
identify(x, y, labels=row.names(mydata)) # identify points  
coords <- locator(type="l") # add lines  
coords # display list
```

Tabellen für Publikationen

```
library(stargazer)  
stargazer(attitude)
```

Tabellen mit dem R-Paket knitr

```
library(knitr)  
kable(head(iris), format = "latex")  
  
kable(head(women), format='latex', booktabs=TRUE)
```

height	weight
58	115
59	117
60	120
61	123
62	126
63	129

Figure 38: pic

Nichtlineare Regression

Folien zum Workshop:

<https://github.com/Japhilko/npRegression/tree/master/slides>

```
library(splines)
```

Beispiel einer interaktiven Karte

interaktiven Karte und Rcode um eine interaktive Karte mit leaflet zu erzeugen.