

R für die Sozialwissenschaften - Teil 1

Jan-Philipp Kolb

22 Juli, 2017

Einführung und Motivation

Pluspunkte von R

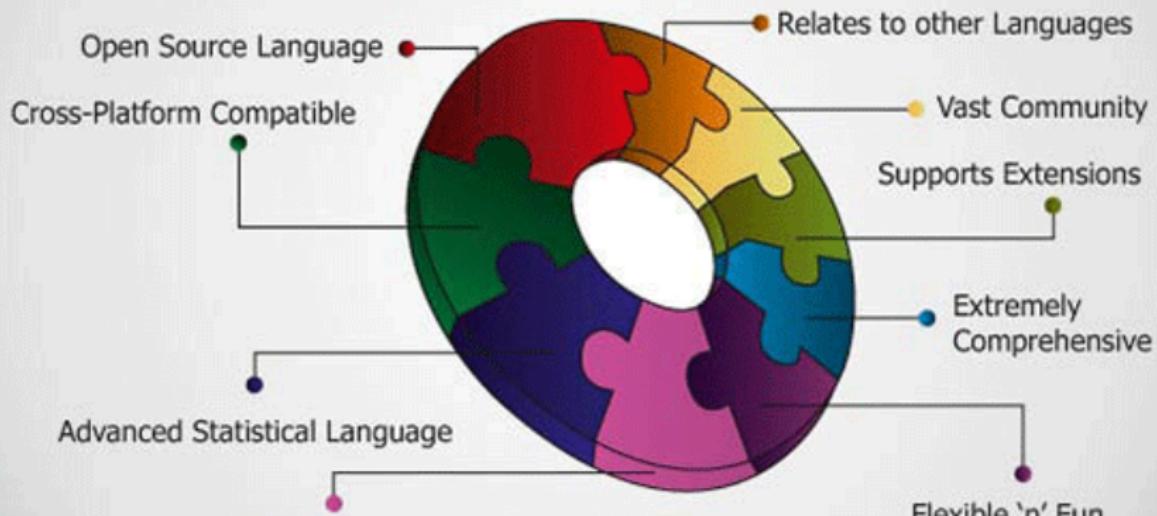
- Als Weg kreativ zu sein ...
- Graphiken, Graphiken, Graphiken
- In Kombination mit anderen Programmen nutzbar
- Zur Verbindung von Datenstrukturen
- Zum Automatisieren
- Um die Intelligenz anderer Leute zu nutzen ;-)
- ...

Gründe

- R ist frei verfügbar. Es kann umsonst runtergeladen werden.
- R ist eine Skriptsprache / Popularität von R

Why Learn R?

edureka!



Ein Hauptgrund - die Community

Revolutions

Daily news about using open source R for big data analysis, predictive modeling, data science, and visualization since 2008

[« Interactive R visuals in Power BI](#) | [Main](#) | [Because it's Friday: Mario in the Park »](#)

June 23, 2017

The R community is one of R's best features

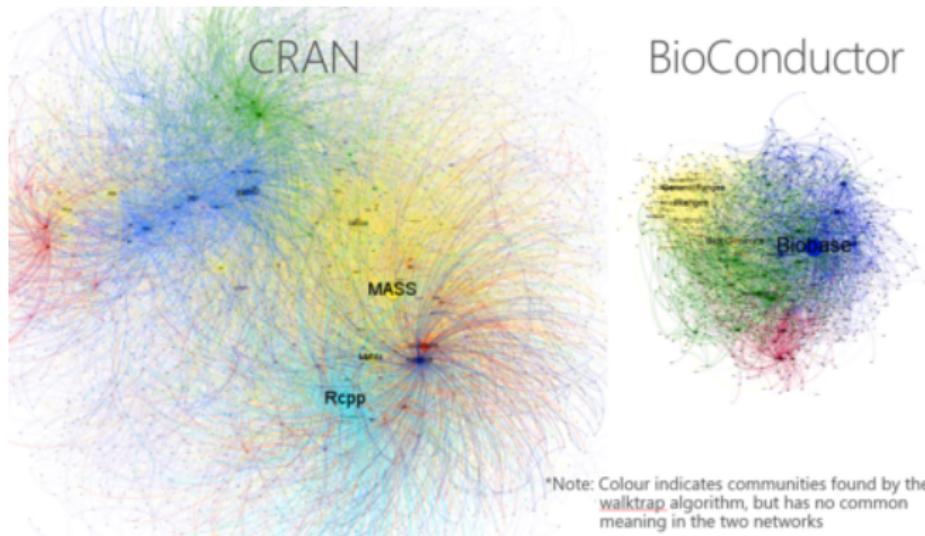
R is incredible software for statistics and data science. But while the bits and bytes of software are an essential component of its usefulness, software needs a **community** to be successful. And that's an area where R really shines, as Shannon Ellis explains in this [lovely ROpenSci blog post](#). For software, a thriving community offers developers, expertise, collaborators, writers and documentation, testers, agitators (to keep the community *and* software on track!), and so much more. Shannon provides links where you can find all of this in the R community:

- **#rstats hashtag** — a responsive, welcoming, and inclusive community of R users to interact with on Twitter
- **R-Ladies** — a world-wide organization focused on promoting gender diversity within the R community, with more than 30 local chapters
- **Local R meetup groups** — a google search may show that there's one in your area! If not, maybe consider starting one! Face-to-face meet-ups for users of all levels are incredibly valuable
- **Rweekly** — an incredible weekly recap of all things R

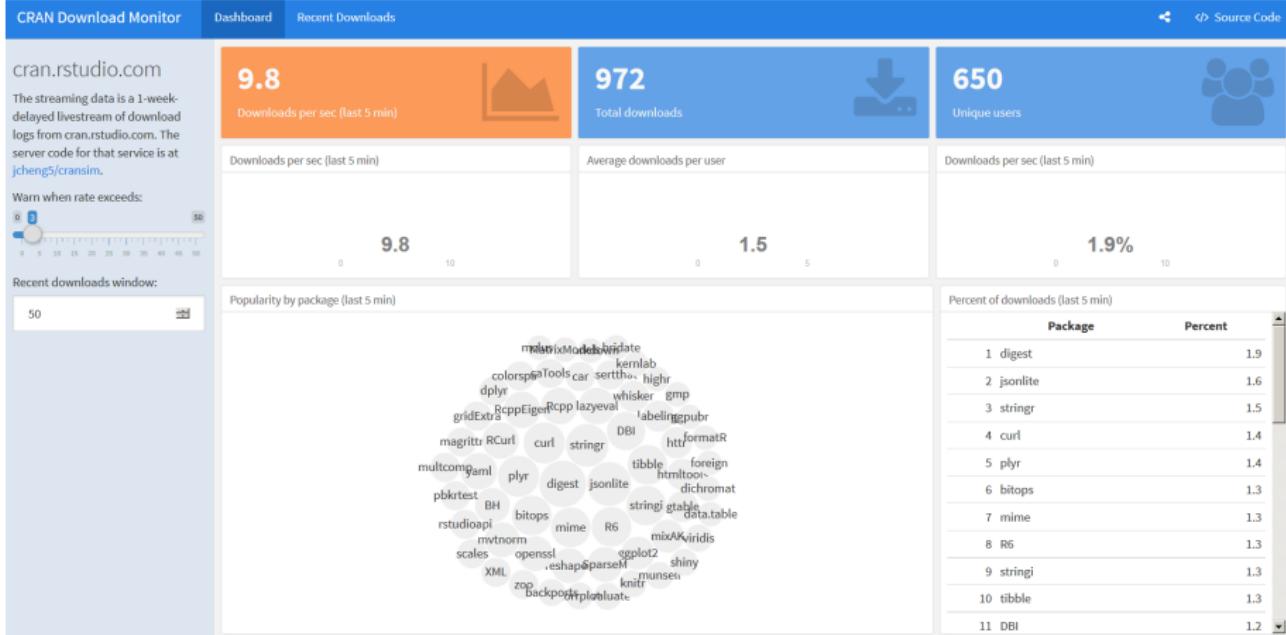
Möglichkeiten auf dem neuesten Stand zu sein

- rweekly
- r-bloggers

Modularer Aufbau



Viel genutzte Pakete



Organisation des Kurses

- Unterlagen sind komplett auf Github hinterlegt, damit man den Kurs gleich mitverfolgen kann (mehr dazu gleich)
- Es werden viele verschiedene kleine Beispieldatensätze verwendet um spezifische Dinge zu zeigen
- Alle Funktionen in R sind mit diesen kleinen Beispielen hinterlegt
- An geeigneten Stellen verwende ich auch größere (sozialwissenschaftliche) Datensätze

Dem Kurs folgen

- <http://japhilko.github.io/Rinter/>

Rinter

Einführungsworkshop
in R für
Sozialwissenschaftler

 View On GitHub

This project is
maintained by [Japhilko](#)

Hosted on [GitHub Pages](#)
using the Dinky theme

Gliederung

Einleitung

- Einführung und Motivation
- Erste Schritte mit R
- Wie bekommt man Hilfe?
- Modularer Aufbau
- Datenimport
- Datenaufbereitung
- Datenexport

Liebe auf den ersten Plot – Grafiken und Datenanalyse mit R

- Basisgrafiken
- Datenanalyse

Das Wiki zum Kurs

- <https://github.com/Japhilko/Rinter/wiki>

This repository Search Pull requests Issues Marketplace Gist

Japhilko / Rinter Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 1 Wiki Settings Insights

ersteSchritte

Jan-Philipp Kolb edited this page an hour ago · 1 revision

Den Datensatz laden

```
## Warning: NAs introduced by coercion

library(foreign)
dat <- read.dta(
  "https://github.com/Japhilko/RSocialScience/blob/master/data/
  GPanel.dta?raw=true")
dat$bazq020a <- as.numeric(dat$bazq020a)
```

Pages 23

Find a Page...

Home

Basisgrafiken

Datenanalyse

Datenaufbereitung

Komplette Foliensätze

Die kompletten Foliensätze kann man hier herunterladen:

- Teil 1 - Von der Einführung bis Graphiken mit lattice
- Teil 2 - Von den Paketen ggplot2 und ggmap bis zu Mehrebenenmodellen
- Teil 3 - Präsentationen, Dashboards, Notebooks und Interaktivität

Der R-code

- Den R-code kann man direkt in die R-Konsole kopieren und ausführen.
- Begleitend zu den Folien wird meistens auch jeweils ein R-File angeboten.
- Der R-code befindet sich in folgendem Ordner:

<https://github.com/Japhilko/RInter/tree/master/code>

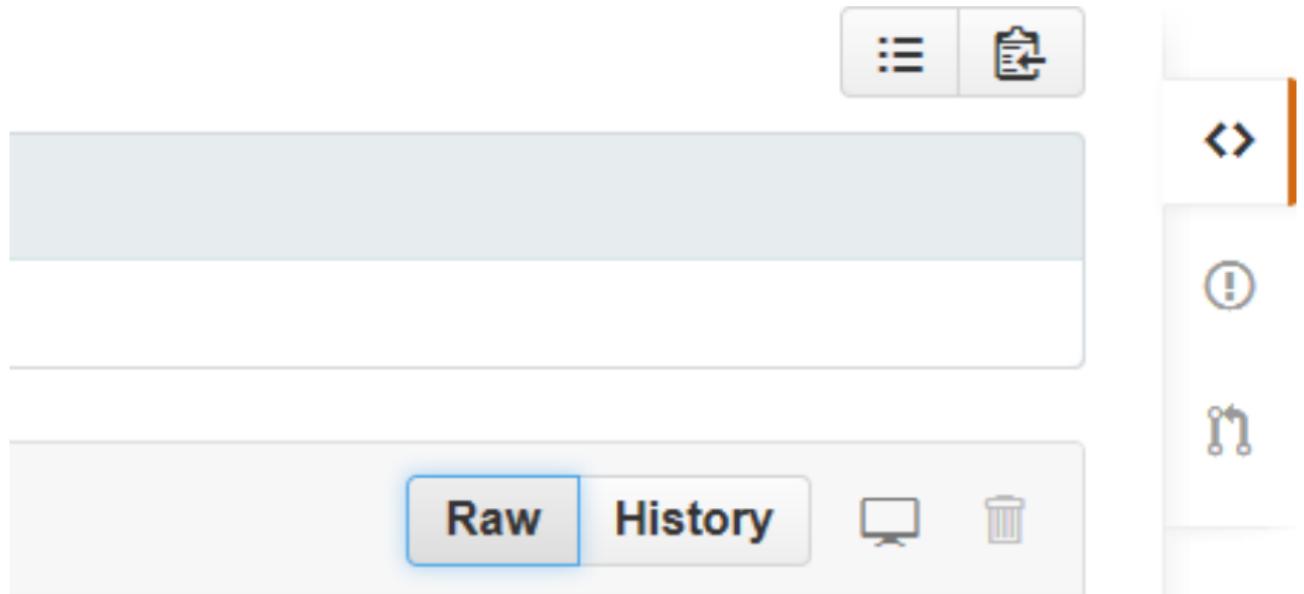
Daten herunterladen

- Vereinzelt sind auch Datensätze vorhanden.
- .csv Dateien können direkt von R eingelesen werden (wie das geht, werde ich noch zeigen).
- Wenn die .csv Dateien heruntergeladen werden sollen - den Raw Button verwenden.
- Alle anderen Dateien (bspw. .RData) auch mittels Raw Button herunterladen.

Ausdrucken

- Zum Ausdrucken eignen sich die pdf-Dateien am besten.
- Diese können mit dem Raw Button heruntergeladen werden.

Raw Button bei Github



Basis R . . .

- Wenn man nur R herunterlädt und installiert, sieht das so aus:
- So habe ich bis 2012 mit R gearbeitet.

R version 2.8.0 (2008-10-28)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-8

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

```
> plot(c(1,2,3))
>
```

Quartz 2 [*]

```
c(1,2,3)
Index
```

```
arm (Version 1.1-16, built: 2008-9-24)
Working directory is /Users/Rense/Documents/RWork
options( digits = 2 )
Loading required package: car
Loading required package: foreign

Attaching package: 'arm'

The following object(s) are masked from package:codetools :

traceplot

> dat.stijn <- read.table("~/Users/Rense/Documents/RWork/stijndata.txt")
> names(dat.stijn) <- c("cons", "cntrywave", "aantal", "resprn", "wave",
+ "country", "sex", "agec", "agesq", "educ",
+ "married", "cohabitend", "separate", "widow", "single",
+ "catb", "prot", "other", "none", "attendance",
+ "volund", "mlavtend", "gdpnormal", "gastil_r", "cwave",
+ "numcountry", "na1", "na2", "na3", "na4")
>
> model.stijn <- glmer(
+ volund ~ sex + educ + agec + agesq + cohabit + separate + widow + single + catb +
+ prot + other + attendance + mlavtend + gdpnormal + gastil_r + (attendance | cwave) +
+ (attendance | numcountry),
+ family="binomial",
+ data=dat.stijn)
>
> for(i in 1:96)
+ {
+ +
+ output <- HI.influence(model.stijn, "cwave", select=i, count=TRUE)
+ save(output, file=paste("~/Users/Rense/Documents/Sociologie/Research Master/Diagnostics.ME/Influence.ME/Testing on Stijn/output ", i, ".RData", sep=""))
+ }
```

... und Rstudio

- Rstudio bietet Heute sehr viel Unterstützung:
- und macht einige Themen dieses Workshops erst möglich

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Project, Workspace, Plots, Tools, and Help. The left pane contains two open files: 'diamondPricing.R' and 'formatPlot.R'. The 'diamondPricing.R' file contains R code for data analysis and visualization. The 'formatPlot.R' file contains a function definition. The bottom-left pane is the 'Console' window, which displays the output of running the R code, including summary statistics for the 'diamonds' dataset and the creation of a scatter plot titled 'Diamond Pricing'. The right pane is the 'Plots' window, showing a scatter plot of Price versus Carat weight, where points are colored by Clarity. A legend on the right side of the plot identifies the clarity levels: I1 (red), SI2 (orange), SI1 (yellow-green), VS2 (green), VS1 (light green), VVS2 (cyan), VVS1 (light blue), and IF (pink).

```
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
avesize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

```
Min. : 0.000  Min. : 0.000  Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean   : 5.731 Mean   : 5.735 Mean   : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max.   :10.740 Max.   :58.900 Max.   :31.800
```

```
summary(diamonds$price)
Min. 1st qu. Median 3rd qu. Max.
326    950    2401   3933   5324  18820
```

```
> avesize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
```

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.

```
## [1] "Sat Jul 22 11:41:10 2017"

## R version 3.3.3 (2017-03-06)
## 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
```

Erste Schritte mit R

R ist eine Objekt-orientierte Sprache

Vektoren und Zuweisungen

- R ist eine Objekt-orientierte Sprache
- <- ist der Zuweisungsoperator (Shortcut: "Alt" + "-")

```
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 - Datentypen

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

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

- mehr zu den möglichen Datentypen später

Funktionen im base-Paket

Funktion	Bedeutung	Beispiel
<code>length()</code>	Länge	<code>length(b)</code>
<code>max()</code>	Maximum	<code>max(b)</code>
<code>min()</code>	Minimum	<code>min(b)</code>
<code>sd()</code>	Standardabweichung	<code>sd(b)</code>
<code>var()</code>	Varianz	<code>var(b)</code>
<code>mean()</code>	Mittelwert	<code>mean(b)</code>
<code>median()</code>	Median	<code>median(b)</code>

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

Aufgabe - Zuweisungen und Funktionen

Erzeugt einen Vektor b mit den Zahlen von 1 bis 5 und berechnet...

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

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 Programmpaket 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.

```
str(fac)
```

```
## Factor w/ 2 levels "1","2": 1 2
```

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]
```

Logische Operatoren

```
# Ist 1 größer als 2?
```

```
1>2
```

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

```
1<2
```

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

```
1==2
```

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

Sequenzen

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

```
1:10
```

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

```
# das gleiche Ergebnis
```

```
seq(1,10)
```

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

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)
```

```
a
```

```
## [1] 3.000000 3.818182 4.636364 5.454545 6.272727 7.0
```

```
## [8] 8.727273 9.545455 10.363636 11.181818 12.000000
```

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

```
b
```

```
## [1] 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0
```

Reihenfolge von Argumenten

1:10

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

```
seq(1,10,1)
```

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

```
seq(length=10,from=1,by=1)
```

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

Wiederholungen

```
# wiederhole 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"
```

- Ein weiteres Beispiel:

```
paste0("A", 1:6)
```

```
## [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 `? name` um Hilfe zu bekommen.

`?mean`

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

`example(lm)`

Vignetten

- Dokumente zur Veranschaulichung und Erläuterung von Funktionen im Paket

`browseVignettes()`

Demos

- zu manchem Paketen gibt es Demonstrationen, wie der Code zu verwenden ist

```
demo()
```

```
demo(nlm)
```

Die Funktion apropos

- sucht alles, was mit dem eingegebenen String in Verbindung steht

```
apropos("lm")
```

```
## [1] ".__C__anova.glm"      ".__C__anova.glm.null" ".__C__g  
## [4] ".__C__glm.null"       ".__C__lm"                 ".__C__m  
## [7] ".__C__optionalMethod" ".colMeans"              ".lm.fit"  
## [10] "colMeans"               "confint.lm"             "contr.h  
## [13] "dummy.coef.lm"         "getAllMethods"        "glm"  
## [16] "glm.control"           "glm.fit"                "KalmanF  
## [19] "KalmanLike"             "KalmanRun"             "KalmanS  
## [22] "kappa.lm"               "lm"                    "lm.fit"  
## [25] "lm.influence"          "lm.wfit"               "model.m  
## [28] "nlm"                   "nlminb"                "predict  
## [31] "predict.lm"              "residuals.glm"        "residua  
## [34] "summary.glm"            "summary.lm"
```

Suchmaschine für die R-Seite

```
RSiteSearch("glm")
```

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, es gibt aber viele Diskussionen zu R
- Sehr detaillierte Diskussionen

The screenshot shows the Stackoverflow homepage. 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 right, there are links for Log In and Sign Up.

Below the navigation, a section titled "Tagged Questions" is shown. It includes a search bar with "[r]" and filters for info, newest, featured (selected), frequent, votes, active, and unanswered. To the right, there's a button to "Ask Question".

A main content area displays a question titled "How to make a great R reproducible example?". The question has 1776 votes, 22 answers (highlighted in green), and 147k views. It includes tags for r and r-faq. The question text discusses the importance of reproducibility in R. Below the question, there are links for community wiki, 11 revs, 8 users 54%, and Hack-R.

To the right, there's a sidebar for the "R Language" documentation, which has 22,187 frequent questions tagged. It includes a link to "about »" and a "Find a request to handle or browse 121 topics." button.

At the bottom, there's a "Related Tags" section with links for ggplot2 (x 2875), dataframe (x 1351), and plot (x 1105).

Quick R

- Immer eine Seite mit Beispielen und Hilfe zu einem Thema
- Beispiel: Quick R - Getting Help

Weitere Links

- Übersicht - Hilfe bekommen in R
- Eine Liste mit HowTo's
- Eine Liste der wichtigsten R-Befehle

Ein Schummelzettel - Cheatsheet

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

Base R Cheat Sheet

Getting Help

Accessing the help files

Print
Get help of a particular function
`help("weighted.mean")`
Search for help on a word or phrase.
`help(package = "dplyr")`
Find help for a package

More about an object

`str(x)`
Get a summary of an object's structure
`class(x)`
Find the class of an object belonging to

Using Libraries

Install, `install.packages("dplyr")`
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(titanic)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd("C:/Users/parts")
Change the current working directory.

Use `projdir` in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors
<code>c(1, 4, 9)</code> A vector of integers
<code>1:6</code> An integer sequence
<code>seq(0, 1, length=10)</code> A complex sequence
<code>rep(1:3, times=2)</code> Repeat a vector
<code>rep(1:3, each=2)</code> Repeat elements of a vector

Vector Functions

sort(x)	rev(x)
Sorts a vector	Returns a reversed vector
<code>table(x)</code>	Counts elements

Selecting Vector Elements

By Position
<code>x[4]</code> The fourth element.
<code>x[1:4]</code> All but the fourth.
<code>x[2:4]</code> Elements two-to-four.
<code>x[-1:4]</code> All elements except first to four.
<code>x[c(1, 5)]</code> Elements one and five.
By Value
<code>x[x == 10]</code> Elements which are equal to 10.
<code>x[x < 0]</code> All elements less than zero.
<code>x[x %in% c(1, 2, 5)]</code> Elements in the set 1, 2, 5.
Named Vectors
<code>x["apple"]</code> Element with name "apple".

Programming

For Loop
<code>for (i in 1:6) { do something }</code>

Example

```
for (i in 1:6) {  
  print(i)  
}
```

While Loop
<code>while (somecond) { do something }</code>

Example

```
while (i < 10) {  
  print(i)  
  i = i + 1  
}
```

If Statements

if (condition){ do something } else if (do something different }
<code>if (i < 10) { print("i is") } else { print("i is not") }</code>

Example

```
if (i < 10) {  
  print("i is")  
} else {  
  print("i is not")  
}
```

Functions

function(x, ...){ do something return(something) }
<code>square = function(x){ squared = x*x return(squared) }</code>

Example

```
square = function(x){  
  squared = x*x  
  return(squared)  
}
```

Reading and Writing Data

Read	Write	Description
<code>df = read.table("titanic.csv")</code>	<code>write.table(df, "titanic.csv")</code>	Read and write a delimited file.
<code>df = read.csv("titanic.csv")</code>	<code>write.csv(df, "titanic.csv")</code>	Read and write a matrix represented via a CSV (a special case of most tables).
<code>load("titanic.RData")</code>	<code>save(df, file = "titanic.RData")</code>	Read and write an R object file.

Comparison Operators

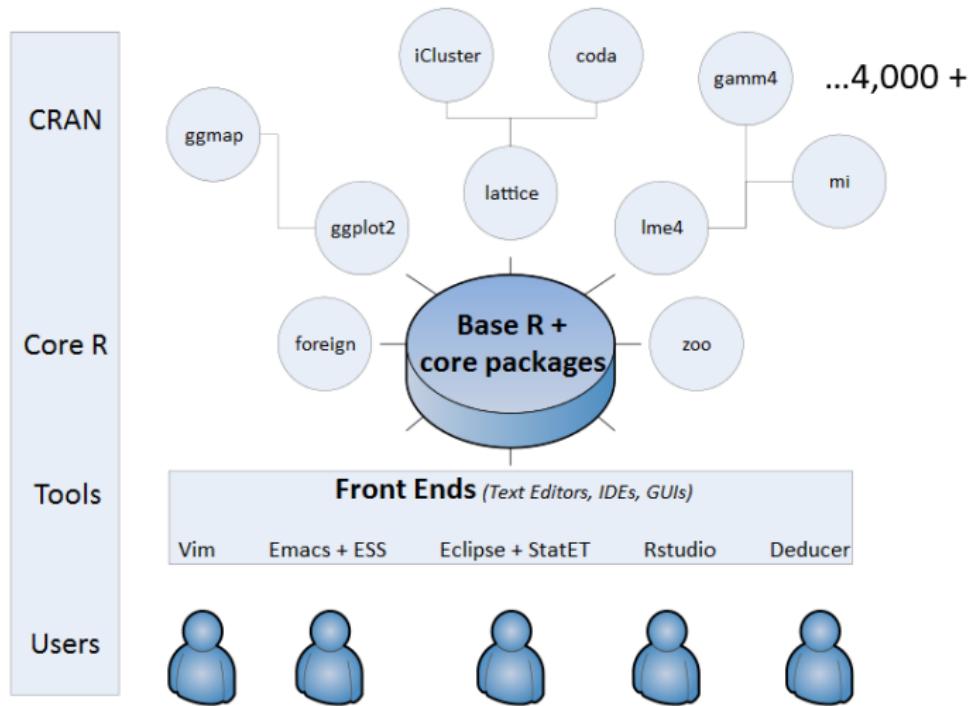
<code>a == b</code>	<code>!a == b</code>	<code>a > b</code>	<code>a < b</code>	<code>isTRUE(a == b)</code>	<code>isTRUE(!a == b)</code>	<code>isTRUE(a > b)</code>	<code>isTRUE(a < b)</code>
<code>==</code>	<code>!=</code>	<code>></code>	<code><</code>	<code>TRUE</code>	<code>FALSE</code>	<code>TRUE</code>	<code>FALSE</code>

Modularer Aufbau

Wo sind die Routinen enthalten

- 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

Übersicht R-Pakete



Installation

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

```
library(lme4)
```

Installation von Paketen mit RStudio

The screenshot shows the RStudio interface. The top-left pane contains a code editor with the file 'paths.R' open, showing the command `setwd("D:/Projekte/Rpackages/germanwebr/Rfunctions")`. The top-right pane is the 'Environment' view, which displays the message 'Environment is empty'. The bottom-right pane is the 'Packages' view, listing various R packages with their versions and download links. The bottom-left pane is the 'Console' view, displaying a welcome message about R being a collaborative project.

Packaging	Description	Version
AER	Applied Econometrics with R	1.2-2
arules	Mining Association Rules and Frequent Itemsets	1.1-2
bitops	Bitwise Operations	1.0-6
boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-11
brew	Templating Framework for Report Generation	1.0-6
car	Companion to Applied Regression	2.0-19
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17
class	Functions for Classification	7.3-10
cluster	Cluster Analysis Extended Rousseeuw et al.	1.15.2
codetools	Code Analysis Tools for R	0.2-8
colorspace	Color Space Manipulation	1.2-4
compiler	The R Compiler Package	3.1.0
DAAG	Data Analysis And Graphics data and functions	1.18

Vorhandene Pakete und Installation

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. Below the navigation bar, there are buttons for 'Install Packages', 'Check for Updates', and a search icon. The main area displays a list of installed packages with their details:

Package	Description	Version	Status
AER	Applied Econometrics with R	1.2-2	
arules	Mining Association Rules and Frequent Itemsets	1.1-2	
bitops	Bitwise Operations	1.0-6	
boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-11	
brew	Templating Framework for Report Generation	1.0-6	

Übersicht viele nützliche Pakete:

- Luhmann - Tabelle mit vielen nützlichen Paketen

Weitere interessante Pakete:

- Paket für den Import/Export - foreign
- Pakete für Survey Sampling
- xtable Paket für die Integration von Latex und R (xtable Galerie)
- Paket zur Erzeugung von Dummies
- Multivariate Normalverteilung
- Paket für Karten

Pakete installieren

Pakete von CRAN Server installieren

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

Pakete von Bioconductor Server installieren

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("GenomicFeatures", "AnnotationDbi"))
```

Pakete von Github installieren

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

```
install_github("hadley/ggplot2")
```

Wie bekomme ich einen Überblick

- Pakete entdecken, die neulich auf CRAN hochgeladen wurden
- Pakete die in letzter Zeit von CRAN heruntergeladen wurden
- Quick-list nützlicher Pakete
- Beste Pakete für Datenbearbeitung und Analyse
- Die 50 meist genutzten Pakete

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

Aufgabe - Zusatzpakete

Gehen Sie auf <https://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



Dateiformate in R

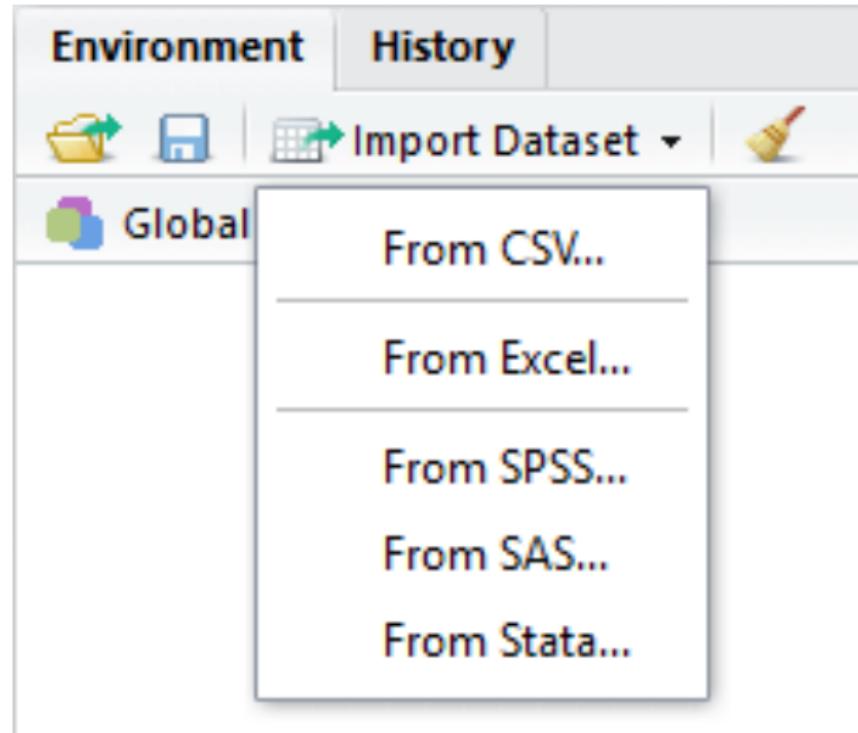
- Von R werden quelloffene, nicht-proprietary Formate bevorzugt
- Es können aber auch Formate von anderen Statistik Software Paketen eingelesen werden
- R-user speichern Objekte gerne in sog. Workspaces ab
- Auch hier jedoch gilt: (fast) alles andere ist möglich

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()`

Datenimport leicht gemacht mit Rstudio



CSV aus dem Web einladen

- Datensatz:

<https://data.montgomerycountymd.gov/api/views/6rqk-pdub/rows.csv?accessType=DOWNLOAD>

- Datenimport mit Rstudio

Import Test Data											
File/URL	<input type="text" value="https://data.montgomerycountymd.gov/api/views/6rqk-pdub/rows.csv?accessType=DOWNLOAD"/> Update										
Data Preview:											
Full Name (character) ^	Gender (character) ^	Current Annual Salary (character) ^	2015 Gross Pay Received (character) ^	2015 Overtime Pay (character) ^	Department (character) ^	Department Name (character) ^	Division (character) ^	Assignment Category (character) ^	Position Title (character) ^	Underfilled Job Title (character) ^	
Aarhus, Pam J.	F	\$68878.16	\$72356.79	NRA	POL	Department of Police	MSB Information Management and Technology Division	Parttime-Regular	Office Services Coordinator	NRA	
Aaron, David J.	M	\$96080.09	\$101857.00	\$4640.99	POL	Department of Police	ISB Major Crimes Division Fugitive Section	Parttime-Regular	Master Police Officer	NRA	
Aaron, Marsha M.	F	\$104196.06	\$103019.73	NRA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Parttime-Regular	Social Worker IV	NRA	
Ababio, Codie A.	M	\$10697.79	\$54181.46	\$4445.15	COR	Correction and Rehabilitation	PRRS Facility and Security	Parttime-Regular	Resident Supervisor II	NRA	
Ababio, Essaysay	M	\$92931.00	\$93468.35	NRA	HCA	Department of Housing and Community Affairs	Single Family Housing Program	Parttime-Regular	Planning Specialist III	NRA	
Abbamonte, Drew B.	M	\$67715.00	\$81392.40	\$10027.11	POL	Department of Police	PSB 6th District Special Assignment Team	Parttime-Regular	Police Officer III	NRA	
Abdelmonem, Marwan M.	M	\$46228.30	\$59663.27	NRA	HHS	Department of Health and Human Services	Head Start	Parttime-Regular	Administrative Specialist II	NRA	
Abdul-Chari, Nasirah J.	F	\$45828.92	\$46783.23	\$6.38	POL	Department of Police	PSB Traffic Division Automated Traffic Enforcement S...	Parttime-Regular	Police Aide	NRA	
Abdullah, Saeed	M	\$61040.57	\$66861.96	\$6569.81	DGS	Department of General Services	Facilities Maintenance	Parttime-Regular	Electrician I	NRA	
Abdur-Rohem, Mikael A.	M	\$56404.00	\$71943.00	\$15342.84	DOT	Department of Transportation	Transit Silver Spring Ride On	Parttime-Regular	Bus Operator	NRA	
Abdele, Hirzah	F	\$151585.60	\$164945.06	NRA	HHS	Department of Health and Human Services	STD and HIV Services	Parttime-Regular	Medical Doctor III - Physician	NRA	
Abdele, Zecharias S.	M	\$44825.99	\$51693.47	\$5240.75	DOT	Department of Transportation	Transit Nicholson Ride On	Parttime-Regular	Bus Operator	NRA	
Abdelin, Amireza	M	\$39062.00	\$450.00	NRA	DOT	Department of Transportation	Transportation Management	Parttime-Regular	Traffic Management Technician II	Traffic Management Technicis	
Abelove, Sherry R.	F	\$93436.50	\$90833.10	NRA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Parttime-Regular	Social Worker III	NRA	
Abera, Joseph M.	M	\$11781.00	\$115786.22	NRA	OTS	Department of Technology Services	EASD - ERP Applications Support	Parttime-Regular	Senior Information Technology Specialist	NRA	
Abi-Jamra, Rania F.	F	\$53009.99	\$46850.36	NRA	LIB	Department of Public Libraries	Olney Library	Parttime-Regular	Library Assistant I	NRA	
Abijamra, Ryan Z.	M	\$17288.00	\$14663.33	NRA	LIB	Department of Public Libraries	Silver Spring Library	Parttime-Regular	Library Desk Assistant	NRA	
Abito, Lydia B.	F	\$40429.58	\$41746.34	\$5500.53	DOT	Department of Transportation	Transit Gaithersburg Ride On	Parttime-Regular	Bus Operator	NRA	
Abikarian, Maral	F	\$20925.51	\$97967.52	\$45.28	POL	Department of Police	PSB Traffic Division School Safety Section	Parttime-Regular	Crossing Guard	NRA	
Abouraya, Nadia L.	F	\$16602.01	\$15592.92	NRA	HHS	Department of Health and Human Services	Community Support Network for People with Disabilities	Parttime-Regular	Office Clerk	NRA	

Previews first 50 entries.

Import Options:

Jan-Philip Kolb

Print Preview

R für die Sozialwissenschaften - Teil 1

22 Juli, 2017

64 / 289

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:

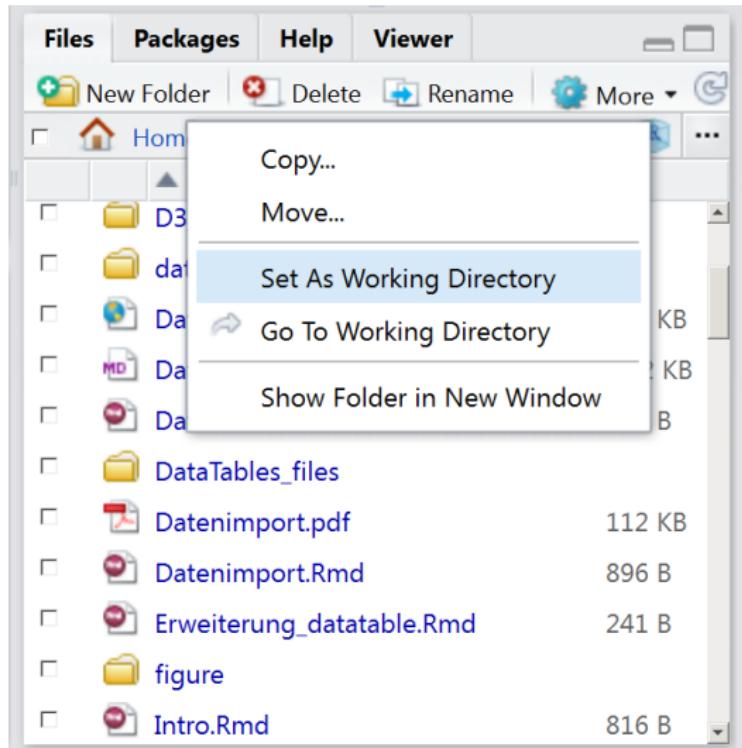
```
main.path <- "C:/" # Beispiel für Windows  
main.path <- "/users/Name/" # Beispiel für Mac  
main.path <- "/home/user/" # Beispiel für Linux
```

Und ändert dann den Pfad mit `setwd()`

`setwd(main.path)`

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

Alternative - Arbeitsspeicher



Die Gesis Panel Daten



gesis



- Gesis Panel Campus File
- Codebuch für die Gesis Panel Daten im DBK

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")
```

Das Paket `readr`

```
install.packages("readr")
```

```
library(readr)
```

- `readr` auf dem Rstudio Blogg

Import von Excel-Daten

- library(readr) ist für den Import von fremden Datenformaten hilfreich
- Wenn Excel-Daten vorliegen - als .csv abspeichern

```
url <- "https://raw.githubusercontent.com/Japhilko/  
GeoData/master/2015/data/whcSites.csv"
```

```
whcSites <- read.csv(url)
```

Der Beispieldatensatz

```
head(data.frame(whcSites$name_en,whcSites$category))

##                                     whcSite
## 1 Cultural Landscape and Archaeological Remains of the Bam
## 2 Minaret and Archaeological Rema
## 3 Historic Centres of Berat and Gj
## 4
## 5 Al Qal'a of E
## 6 M

##   whcSites.category
## 1     Cultural
## 2     Cultural
## 3     Cultural
## 4     Cultural
## 5     Cultural
## 6     Cultural
```

Das Paket haven

Import and Export 'SPSS', 'Stata' and 'SAS' Files

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

- Das R-Paket haven auf dem Rstudio Blogg

SPSS Dateien einlesen

- Zunächst muss wieder der Pfad zum Arbeitsverzeichnis angeben werden.
- SPSS-Dateien können auch direkt aus dem Internet geladen werden:

```
library(haven)
mtcars <- read_sav(
  "https://github.com/Japhilko/RInterfaces/raw/master/
  data/mtcars.sav")
```

foreign kann ebenfalls zum Import genutzt werden

```
library(foreign)
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

```
library(haven)
oecd <- read_dta("https://github.com/Japhilko/IntroR/
                  raw/master/2017/data/oecd.dta")
```

- Einführung in Import mit R (is.R)

Das Paket `readstata13`

```
# install.packages("readstata13")  
  
library(readstata13)  
?read.dta13
```

Das Paket rio

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

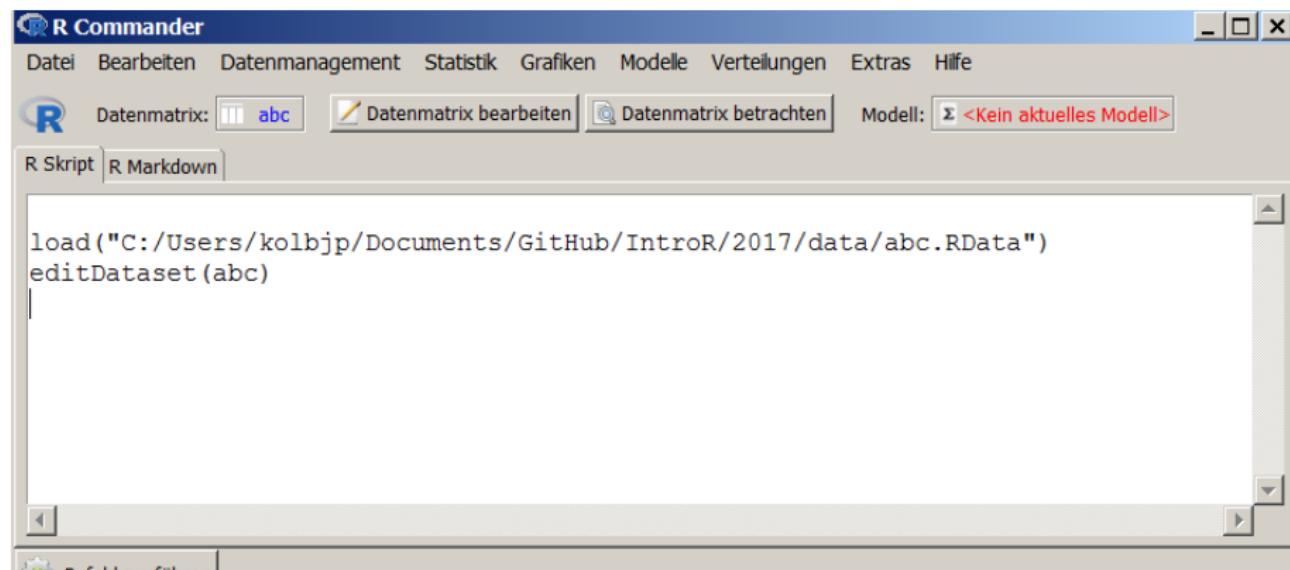
- rio: A Swiss-Army Knife for Data I/O

Weitere Alternative Rcmdr

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

- Funktioniert auch mit Rstudio

```
library(Rcmdr)
```



Aufgabe

- Gehen Sie auf meine Github Seite

<https://github.com/Japhilko/RSocialScience/tree/master/data>

- Importieren Sie den Datensatz GPanel.dta

Datenaufbereitung

Data Frames

Beispieldaten einlesen:

```
library(foreign)
dat<-read.dta("https://github.com/Japhilko/RSocialScience/
               raw/master/data/GPanel.dta")

typeof(dat)

## [1] "list"
```

In Dataframe übertragen

Diese beiden Vektoren zu einem data.frame verbinden:

```
Daten <- data.frame(dat)
```

Anzahl der Zeilen/Spalten herausfinden

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

```
## [1] 100
```

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

```
## [1] 23
```

Die Daten anschauen

- die ersten Zeilen anschauen

`head(Daten)`

- Übersicht mittels Rstudio

The screenshot shows the RStudio interface with the 'Environment' tab selected. The global environment contains three data objects: 'Chem97', 'dat', and 'Daten'. Each object is listed with its name, size (number of observations and variables), and a small icon.

Object	Description	Icon
Chem97	31022 obs. of 8 variables	Calculator icon
dat	100 obs. of 23 variables	Calculator icon
Daten	100 obs. of 23 variables	Calculator icon

Indizieren

Indizieren eines dataframe:

```
Daten[1,1]
```

```
## [1] Eher zufrieden
```

```
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
Daten[2,]
```

```
##           a11c019a           a11c020a           a11c021a
```

```
## 2 Sehr zufrieden Eher unzufrieden Eher unzufrieden Stimme e
```

```
##           a11c023a           a11c024a
```

```
## 2 Stimme eher zu Stimme voll und ganz zu Eher niedrigeren I
```

```
##           a11c026a           a11c027a a11c028a a1
```

```
## 2 Mehrmals die Woche Mindestens einmal im Monat Täglich T
```

```
##           a11c031a           a11c032a a11c033a
```

```
## 2 Mindestens einmal im Monat Mehrmals im Monat Seltener
```

```
##
```

Operatoren um Subset für Datensatz zu bekommen

Diese Operatoren eignen sich gut um Datensätze einzuschränken

```
Dauer <- as.numeric(Daten$bazq020a)  
head(Daten[Dauer>20,])
```

```
##           a11c019a           a11c020a           a11c021a  
## 2  Sehr zufrieden Eher unzufrieden Eher unzufrieden Stimme  
## 3  Eher zufrieden    Sehr zufrieden Eher unzufrieden Stimme  
## 5  Eher zufrieden    Eher zufrieden   Eher zufrieden  
## NA          <NA>          <NA>          <NA>  
## 9  Sehr zufrieden    Eher zufrieden  Sehr zufrieden  
## 15 Sehr zufrieden    Sehr zufrieden  Sehr zufrieden  
##           a11c023a           a11c024a  
## 2      Stimme eher zu Stimme voll und ganz zu  
## 3  Stimme eher nicht zu           Stimme eher zu  
## 5      Stimme eher zu           Stimme eher zu  
## NA          <NA>          <NA>
```

Einschränken mit dem Paket tidyverse

- einfacher geht es mit dem Paket tidyverse

```
library(tidyverse)
filter(Daten, Dauer>20)
```

##	a11c019a	
## 1	Sehr zufrieden	Eher unzufrieden
## 2	Eher zufrieden	Sehr zufrieden
## 3	Eher zufrieden	Eher zufrieden
## 4	Sehr zufrieden	Eher unzufrieden
## 5	Sehr zufrieden	Sehr zufrieden
## 6	Eher zufrieden	Sehr zufrieden
## 7	Sehr zufrieden	Sehr zufrieden
## 8	Sehr zufrieden	Sehr zufrieden
## 9	Eher zufrieden	Eher zufrieden
## 10	Sehr zufrieden	Eher unzufrieden
## 11	Eher zufrieden	Eher zufrieden

Datensätze einschränken

```
SEX <- Daten$a11d054a
```

```
Daten[SEX=="Männlich",]
```

```
##                                     a11c019a  
## 1             Eher zufrieden Weder zufrieden noch unzufrieden  
## 2             Sehr zufrieden Eher unzufrieden  
## 3             Eher zufrieden Sehr zufrieden  
## 4             Eher zufrieden Sehr zufrieden  
## 5             Eher zufrieden Eher zufrieden  
## 7             Eher zufrieden Eher zufrieden  
## 9             Sehr zufrieden Eher zufrieden  
## 12            Sehr zufrieden Eher zufrieden  
## 15            Sehr zufrieden Sehr zufrieden  
## 16            Sehr zufrieden Sehr zufrieden  
## 17 Weder zufrieden noch unzufrieden Eher unzufrieden
```

Weitere wichtige Optionen

Ergebnis in ein Objekt speichern

```
subDat <- Daten[Dauer>20,]
```

mehrere Bedingungen können mit

& verknüpft werden:

```
Daten[Dauer>18 & SEX=="Männlich",]
```

##	a11c019a	
## 2	Sehr zufrieden	Eher unzufrieden
## 3	Eher zufrieden	Sehr zufrieden
## 5	Eher zufrieden	Eher unzufrieden
## 9	Sehr zufrieden	Eher unzufrieden
## 15	Sehr zufrieden	Sehr zufrieden
## 18	Eher zufrieden	Sehr zufrieden
## 20	Eher zufrieden	Eher unzufrieden
## 29	Sehr zufrieden	Sehr zufrieden
## 41	Sehr zufrieden	Eher unzufrieden

Die Nutzung einer Sequenz

Daten[1:3,]

```
##           a11c019a                  a11c020a      a11c021a  
## 1 Eher zufrieden Weder zufrieden noch unzufrieden Sehr zufrieden  
## 2 Sehr zufrieden                      Eher unzufrieden Eher unzufrieden  
## 3 Eher zufrieden                      Sehr zufrieden Eher unzufrieden  
##           a11c022a                  a11c023a      a11c024a  
## 1 Stimme eher nicht zu      Stimme eher zu      Stimme eher zu  
## 2 Stimme eher nicht zu      Stimme eher zu      Stimme voll und ganz zu  
## 3 Stimme eher nicht zu      Stimme eher nicht zu     Stimme voll und ganz zu  
##           a11c025a                  a11c026a      a11c027a  
## 1 Eher niedrigeren Lebensstandard      Seltener  
## 2 Eher niedrigeren Lebensstandard Mehrmals die Woche  
## 3 Denselben Lebensstandard            Täglich  
##           a11c028a a11c029a      a11c030a  
## 1 Mehrmals die Woche   Täglich   Täglich
```

Variablen Labels

```
library(foreign)
dat <- read.dta("https://github.com/Japhilko/RSocialScience/blob/master/data/01.dta")
attributes(dat)

var.labels <- attr(dat, "var.labels")
```

- Genauso funktioniert es auch mit dem Paket haven

```
library(haven)
dat2 <- read_dta(
  "https://github.com/Japhilko/RSocialScience/blob/master/data/01.dta")
var.labels2 <- attr(dat, "var.labels")
```

Die Spaltennamen umbenennen

- Mit `colnames` bekommt man die Spaltennamen angezeigt

```
colnames(dat)
```

```
## [1] "a11c019a" "a11c020a" "a11c021a" "a11c022a" "a11c023a"  
## [7] "a11c025a" "a11c026a" "a11c027a" "a11c028a" "a11c029a"  
## [13] "a11c031a" "a11c032a" "a11c033a" "a11c034a" "bazq020a"  
## [19] "a11d056z" "a11d092a" "a11c100a" "a11c111a" "a11c109a"
```

- So kann man die Spaltennamen umbenennen:

```
colnames(dat) <- var.labels
```

- Analog geht das für die Reihennamen

```
rownames(dat)
```

```
## [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"
```

Indizieren

- Das Dollarzeichen kann man auch nutzen um einzelne Spalten anzusprechen

```
head(dat$a11c019a)
```

```
## [1] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufrieden  
## [5] Eher zufrieden Sehr zufrieden  
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```
dat$a11c019a[1:10]
```

```
## [1] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufrieden  
## [5] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufrieden  
## [9] Sehr zufrieden Sehr zufrieden  
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

Auf Spalten zugreifen

- Wie bereits beschrieben kann man auch Zahlen nutzen um auf die Spalten zuzugreifen

```
head(dat[, 1])
```

```
head(dat[, "a11c019a"]) # liefert das gleiche Ergebnis
```

Exkurs - Labels wie verwenden

Tools for Working with Categorical Variables (Factors)

```
library("forcats")
```

- fct_collapse - um Faktorlevel zusammenzufassen
- fct_count - um die Einträge in einem Faktor zu zählen
- fct_drop - Unbenutzte Levels raus nehmen

Rekodieren

```
library(car)

head(dat$a11c020a)

## [1] Weder zufrieden noch unzufrieden Eher unzufrieden
## [3] Sehr zufrieden                      Sehr zufrieden
## [5] Eher zufrieden                      Eher zufrieden
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht

head(recode(dat$a11c020a, "'Eher unzufrieden'='A';else='B'"))

## [1] B A B B B B
## Levels: A B
```

Das Paket tibble

```
install.packages("tibble")

library(tibble)
gpanel1 <- as_tibble(dat)
gpanel1

## # A tibble: 100 × 23
##       a11c019a      a11c020a
## * <fctr>          <fctr>
## 1 Eher zufrieden Weder zufrieden noch unzufrieden Sehr zufrieden
## 2 Sehr zufrieden
## 3 Eher zufrieden
## 4 Eher zufrieden
## 5 Eher zufrieden
## 6 Sehr zufrieden
## 7 Eher zufrieden
## 8 Eher zufrieden
```

Schleifen

```
erg <- vector()  
  
for (i in 1:ncol(dat)){  
  erg[i] <- length(table(dat[,i]))  
}  
}
```

Fehlende Werte ausschließen

- Mathe-Funktionen haben in der Regel einen Weg, um fehlende Werte in ihren Berechnungen auszuschließen.
- `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` und `'max()'` all take the `na.rm` argument.

Fehlende Werte umkodieren

```
Daten$bazq020a[Daten$bazq020a==99] <- NA
```

- Quick-R zu fehlenden Werten
- Fehlende Werte rekodieren

Mit Strings arbeiten

```
gsub("l","L","Hallo Welt")
```

```
## [1] "HaLLo WeLt"
```

- Natural Language Processing - Tutorial auf der UseR 2017

Weitere Links

- Tidy data - das Paket `tidyverse`
- Die `tidyverse` Sammlung
- Data wrangling with R and RStudio

Datenexport

Die Exportformate von R

- In R werden offene Dateiformate bevorzugt
- Genauso wie `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)
```

		A B
1		A
2		B
3		C
4		D

Überblick Daten Import/Export

- wenn mit R weitergearbeitet wird, eignet sich das .RData Format am Besten:

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

- Der Datensatz kann dann mit load wieder eingelesen werden

```
load("mydata.RData")
```

Daten in .csv Format abspeichern

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

- Wenn mit Deutschem Excel weitergearbeitet werden soll, eignet sich write.csv2 besser

```
write.csv2(mydata, file="mydata.csv")
```

- Sonst sieht das Ergebnis so aus:

	A
1	,"A","B"
2	1,1,"A"
3	2,2,"B"
4	3,3,"C"
5	4,4,"D"
6	

Das Paket xlsx

R xlsx package : A quick start guide to manipulate Excel files in R

AdChoices

Microsoft Excel

Download for Java

Excel Tutorial



- Install and load xlsx package
- Read an Excel file
- Write data to an Excel file

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

Reading/Writing Stata (.dta) files with Foreign

December 4, 2012

By `is.R()`

- Funktionen im Paket `foreign`

R topics documented:

<code>lookup.xport</code>	2
<code>read.arff</code>	3
<code>read.dbf</code>	4
<code>read.dta</code>	5
<code>read.epiinfo</code>	7
<code>read.mtp</code>	8
<code>read.octave</code>	9
<code>read.spss</code>	10
<code>read.ssdi</code>	12

Daten in stata Format abspeichern

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

Das Paket rio

```
install.packages("rio")
```

Import, Export, and Convert Data Files

The idea behind `rio` is to simplify the process of importing data into R and exporting data from R. This process is, probably unnecessarily, extremely complex for beginning R users. Indeed, R supplies [an entire manual](#) describing the process of data import/export. And, despite all of that text, most of the packages described are (to varying degrees) out-of-date. Faster, simpler, packages with fewer dependencies have been created for many of the file types described in that document. `rio` aims to unify data I/O (importing and exporting) into two simple functions: `import()` and `export()` so that beginners (and experienced R users) never have to think twice (or even once) about the best way to read and write R data.

Daten als .sav abspeichern (SPSS)

```
library("rio")
# create file to convert

export(mtcars, "data/mtcars.sav")
```

Dateiformate konvertieren

```
export(mtcars, "data/mtcars.dta")  
  
# convert Stata to SPSS  
convert("data/mtcars.dta", "data/mtcars.sav")
```

Links Export

- Quick R für das Exportieren von Daten:
- Hilfe zum Export auf dem cran Server
- Daten aus R heraus bekommen

Basisgrafiken

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.

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.

Datensatz

```
library(mlmRev)  
data(Chem97)
```

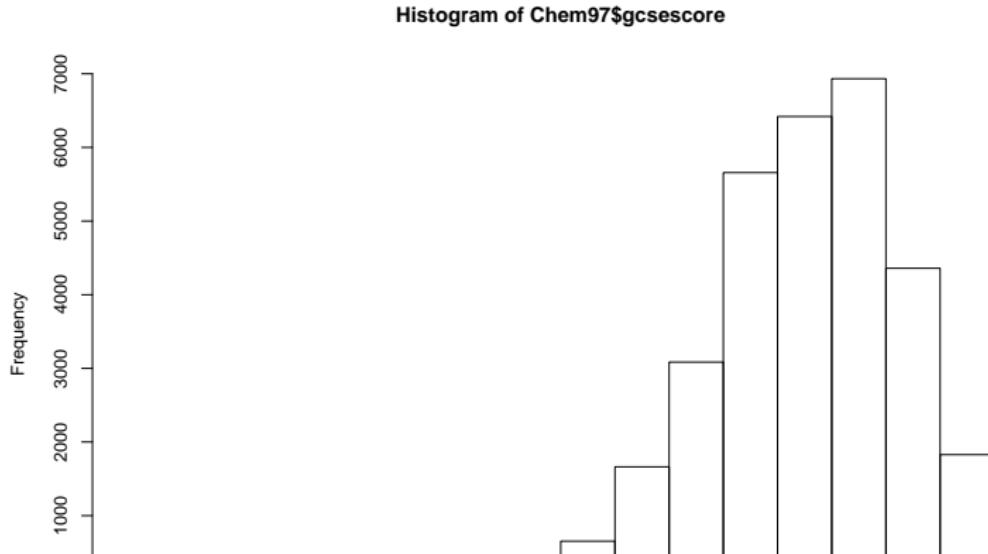
- [lea] Local Education Authority - a factor
- [school] School identifier - a factor
- [student] Student identifier - a factor
- [score] Point score on A-level Chemistry in 1997
- [gender] Student's gender
- [age] Age in month, centred at 222 months or 18.5 years
- [gcsescore] Average GCSE score of individual.
- [gcsecnt] Average GCSE score of individual, centered at mean.

Histogramm - Die Funktion hist()

Wir erstellen ein Histogramm der Variable gcsescore:

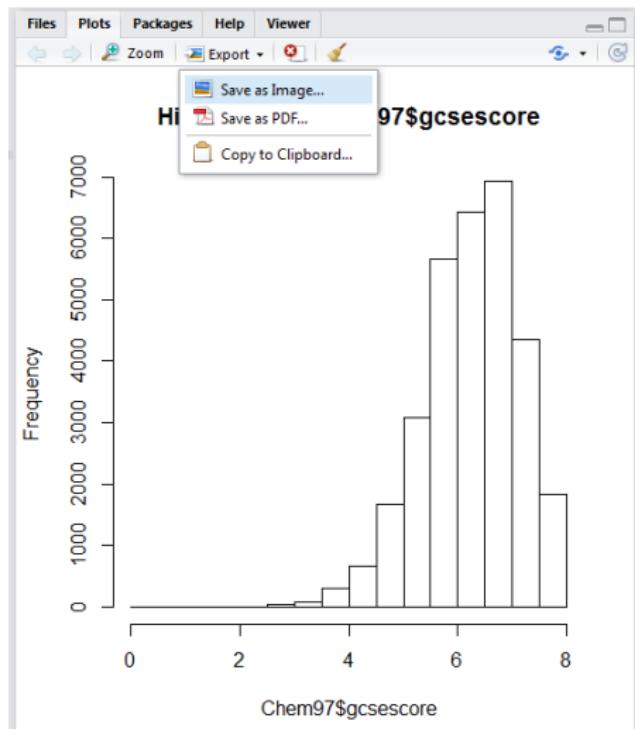
?hist

`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()
```

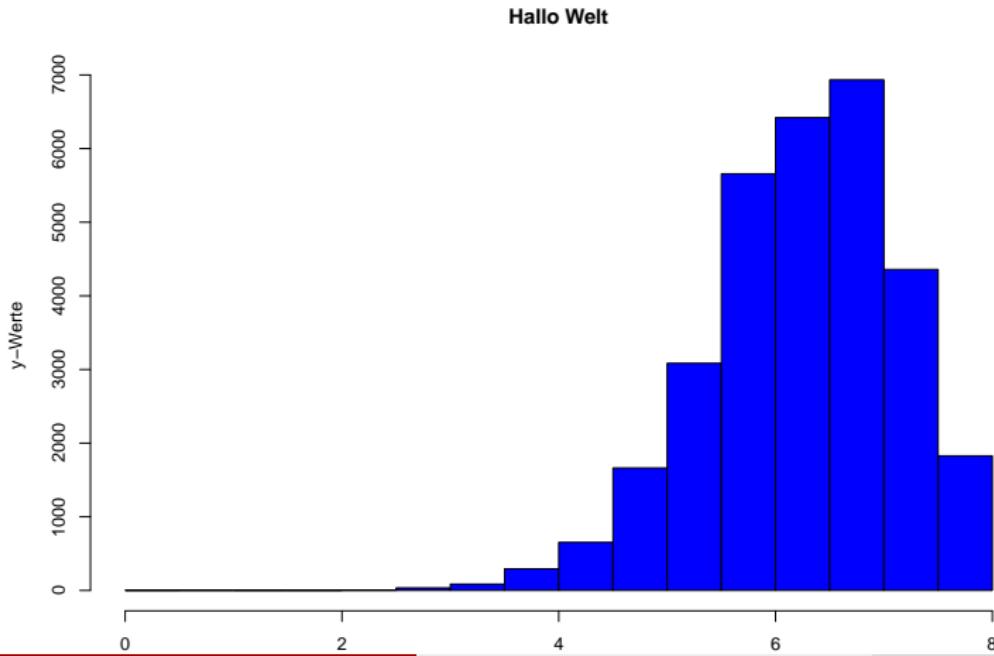
Histogramme

- 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	<code>main="Hallo Welt"</code>
xlab	x-Achsenbeschriftung	<code>xlab="x-Werte"</code>
ylab	y-Achsenbeschriftung	<code>ylab="y-Werte"</code>
col	Farbe	<code>col="blue"</code>

Histogramm

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



Barplot

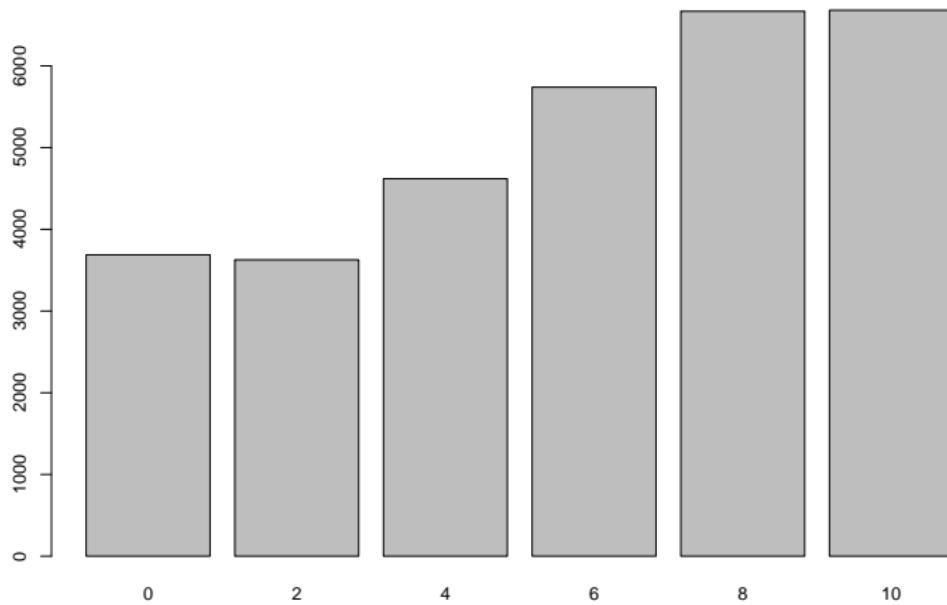
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

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

```
barplot(tabScore)
```

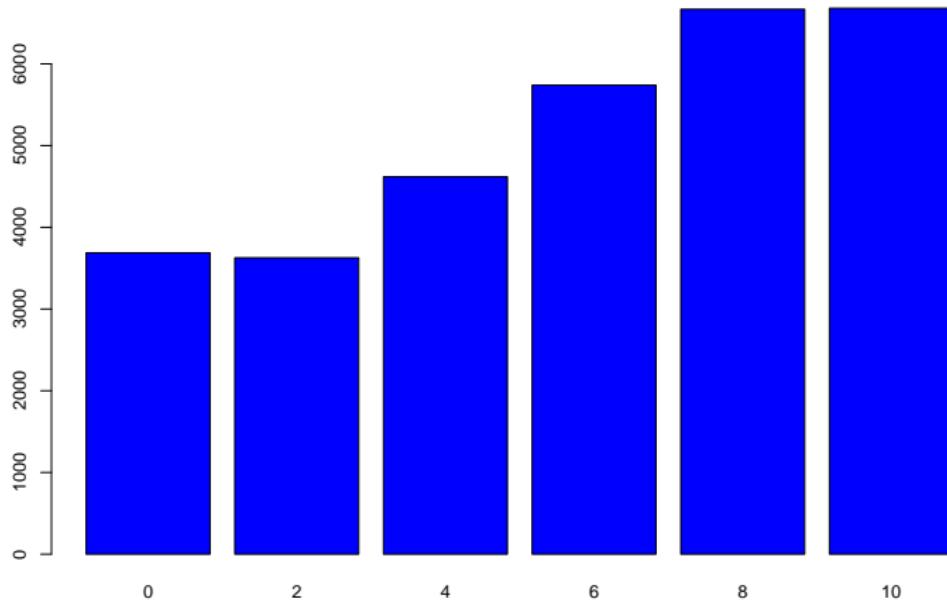
Barplots und barcharts

```
barplot(tabScore)
```



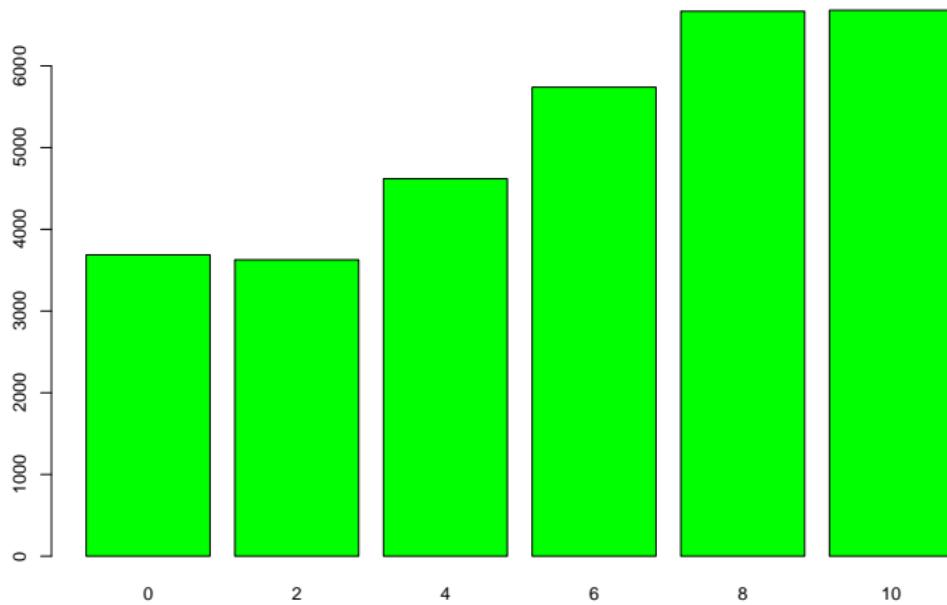
Mehr Farben:

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



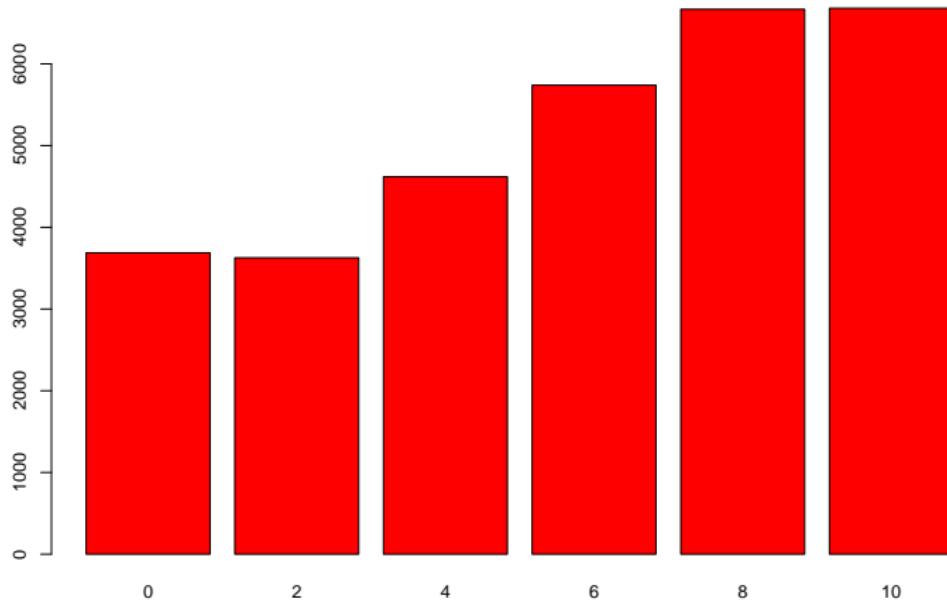
Grüne Farbe

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



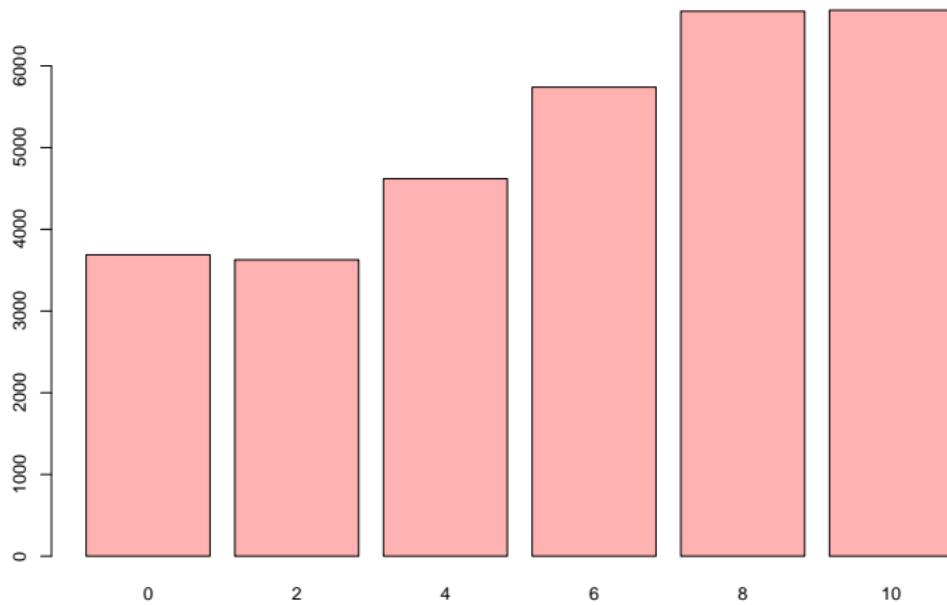
Rote Farbe

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



Transparent

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



Scatterplots

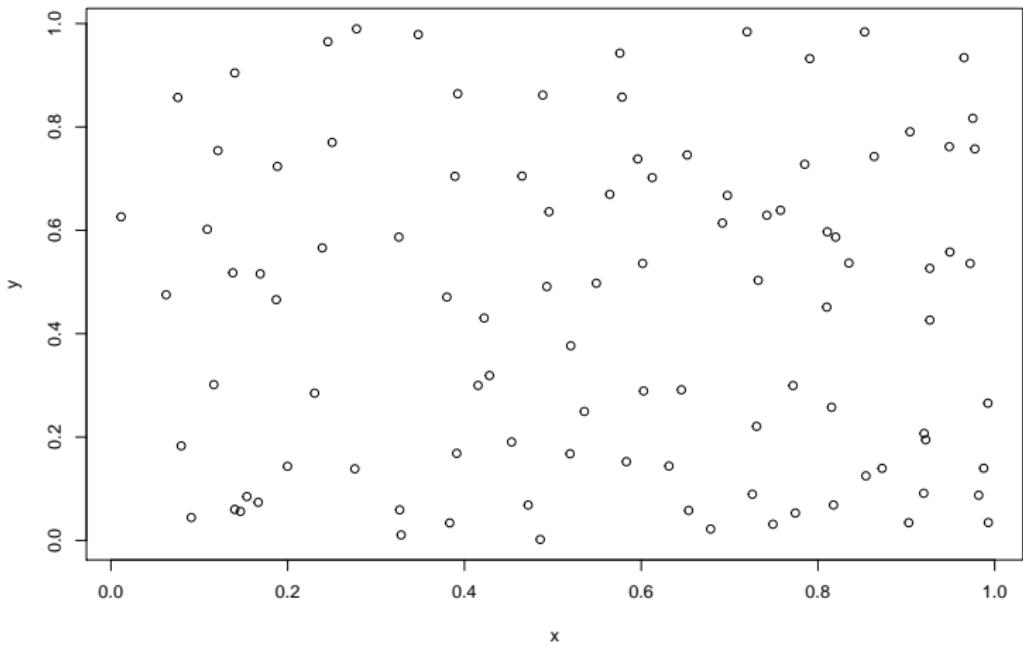
- Ein einfacher two-way Scatterplot kann mit der Funktion `plot()` erstellt werden
- `plot()` muss mindestens ein `x` und ein `y` Beobachtungsvektor übergeben werden
- Um die Farbe der Plot-Symbole anzupassen gibt es die Option `col` (Farbe als character oder numerisch)
- Die Plot-Symbole selbst können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

Beispieldaten für Scatterplot

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

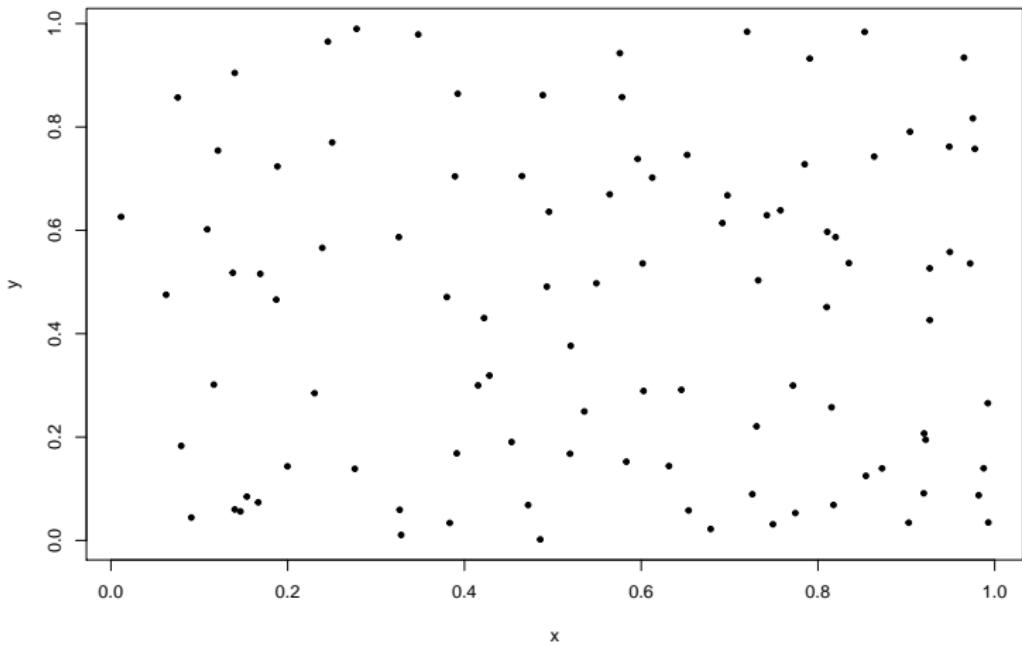
Einfacher Scatterplot

`plot(x, y)`



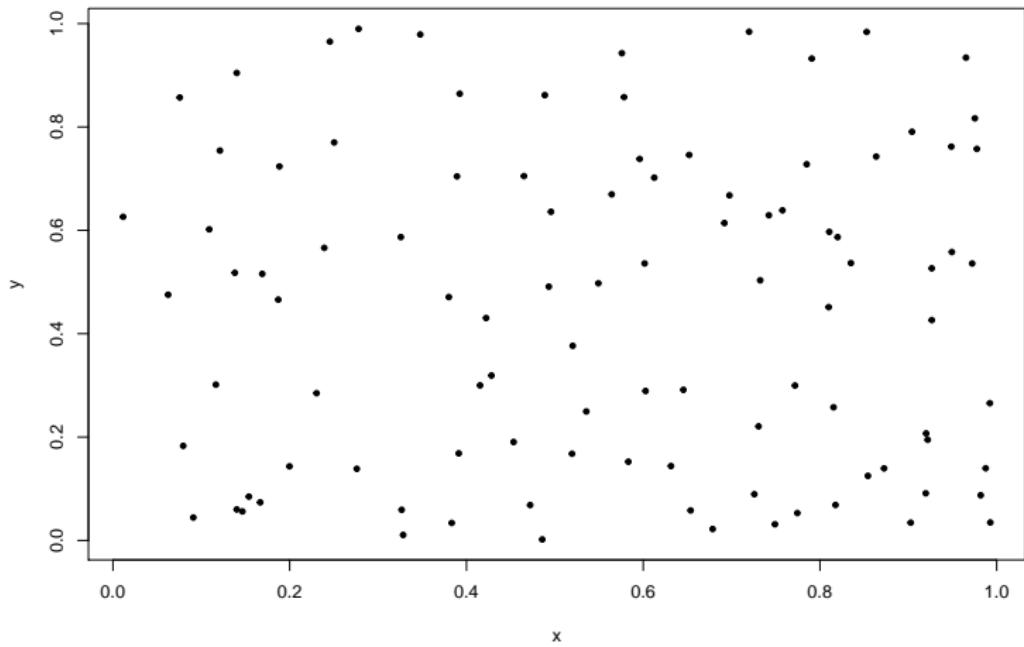
Einfacher Scatterplot II

```
plot(x,y,pch=20)
```



Einfacher Scatterplot III

```
plot(x,y,pch=20)
```



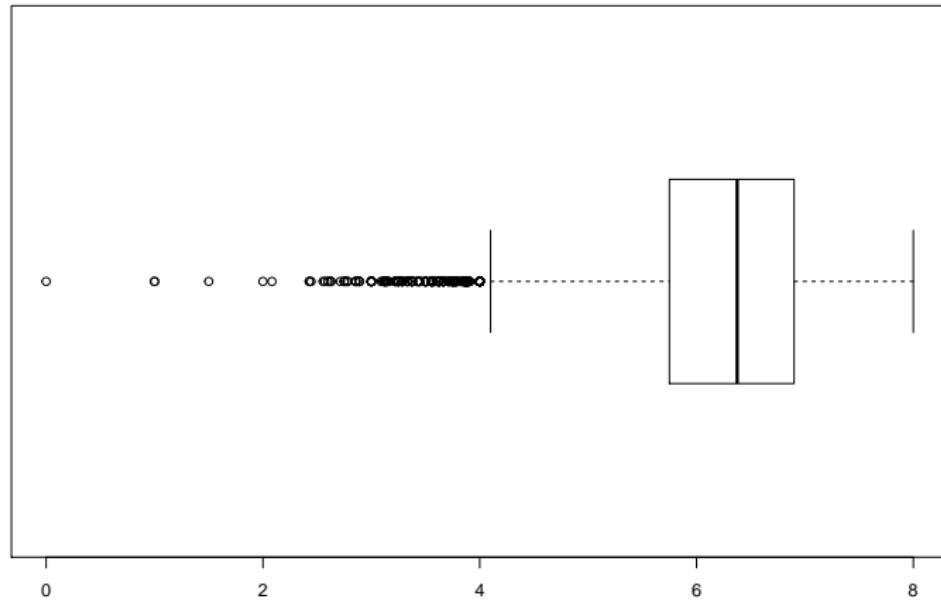
Boxplot

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

?`boxplot`

Horizontaler Boxplot

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

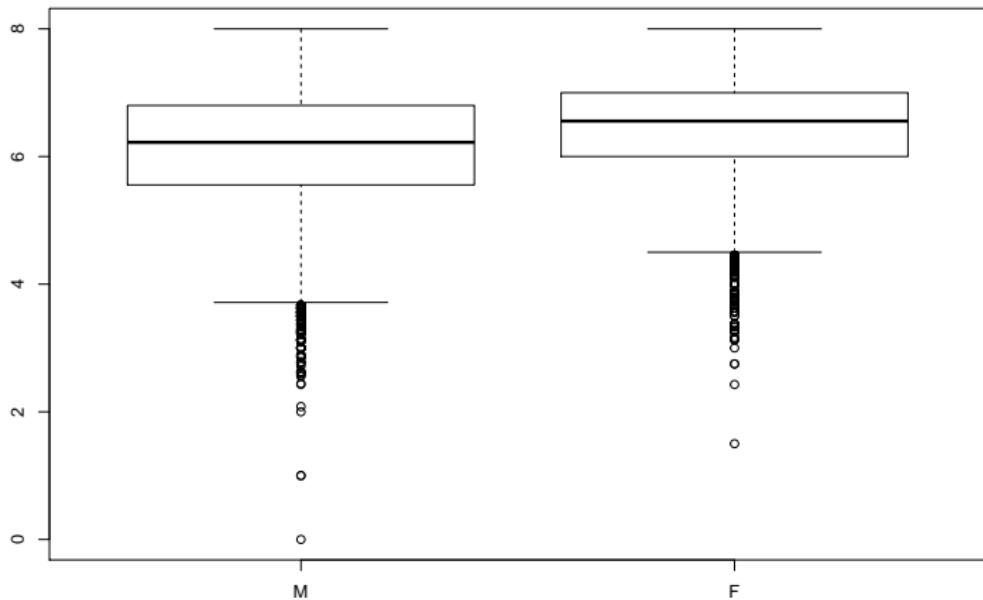


Gruppierte Boxplots

- 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 grupierter Boxplot

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



Alternativen zu Boxplot

Violinplot

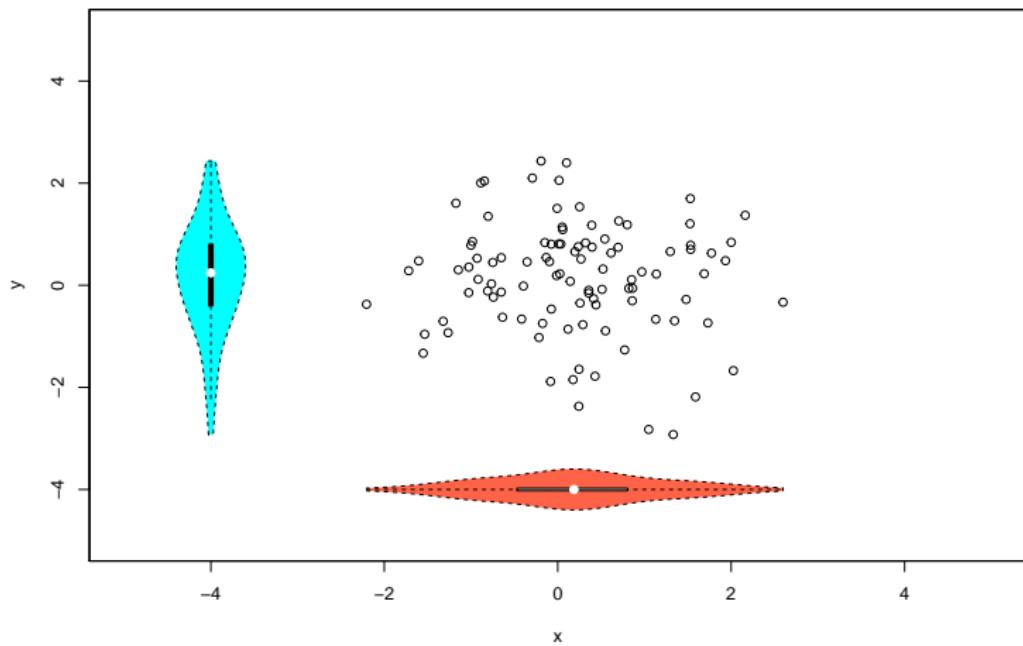
- Baut auf Boxplot auf
- Zusätzlich Informationen über Dichte der Daten
- Dichte wird über Kernel Methode berechnet.
- weißer Punkt - Median
- Je weiter die Ausdehnung, desto größer ist die Dichte an dieser Stelle.

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

Die Bibliothek vioplot

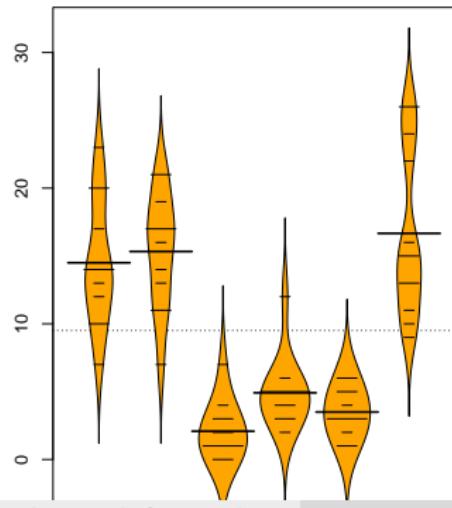
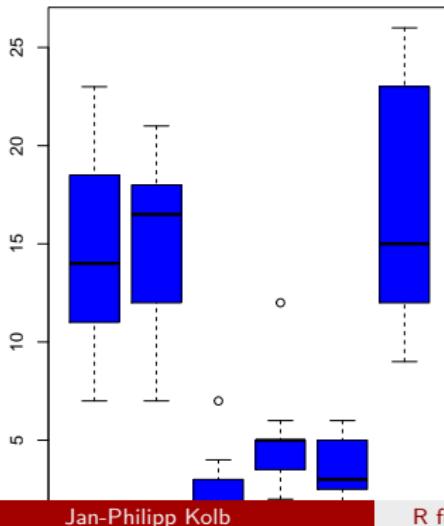
```
library(vioplot)
plot(x, y, xlim=c(-5,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-4,
        add=TRUE, lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=-4,
        add=TRUE, lty=2)
```

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")
```

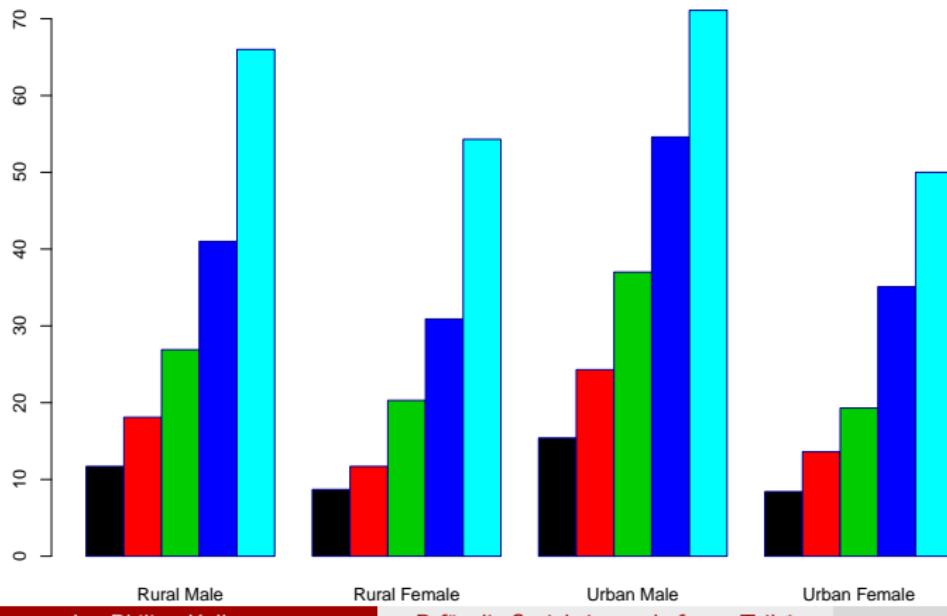


CMYK Farbschema

```
pdf("test.cmyk.pdf", colormodel='cmyk')
pie(1:10, col=1:10)
dev.off()
```

Aufgabe - einfache Grafiken

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



Datenanalyse

Den Datensatz laden

```
library(foreign)
dat <- read.dta(
  "https://github.com/Japhilko/RSocialScience/blob/master/data/
  GPanel.dta?raw=true")
dat$bazq020a <- as.numeric(dat$bazq020a)
```

Streuungsmaße

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

```
var(dat$bazq020a)  
## [1] NA  
  
var(dat$bazq020a,na.rm=T)  
## [1] 476.8859  
  
sd(dat$bazq020a,na.rm=T)  
## [1] 21.83772  
  
range(dat$bazq020a,na.rm=T)
```

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

```
table(dat$a11d054a)
```

```
##  
## Männlich Weiblich  
##          43         57
```

Tabellieren - weiteres Beispiel

```
?table  
  
table(dat$a11d054a)  
  
##  
## Männlich Weiblich  
##      43       57  
  
table(dat$a11d054a,dat$a11d056z)  
  
##  
##           Ambiguous answer Item nonresponse Not reached Un  
##   Männlich                      0                      0          0  
##   Weiblich                      0                      0          0  
##  
##           Not in panel 18 bis unter 20 Jahre 20 bis unter  
##   Männlich                      0                      2  
##   Weiblich
```

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

```
?prop.table
```

```
prop.table(table(dat$a11d054a,dat$a11d056z),1)
```

```
##  
##          Ambiguous answer Item nonresponse Not reached Un  
##  Männlich      0.000000000 0.000000000 0.000000000  
##  Weiblich      0.000000000 0.000000000 0.000000000  
##  
##          Not in panel 18 bis unter 20 Jahre 20 bis unter  
##  Männlich      0.000000000 0.046511630 0.0  
##  Weiblich      0.000000000 0.035087720 0.  
##  
##          25 bis unter 30 Jahre 30 bis unter 35 Jahre  
##  Männlich      0.046511630 0.069767440  
##  Weiblich      0.087719300 0.035087720  
##
```

Die aggregate Funktion

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

```
aggregate(dat$bazq020a, by=list(dat$a11d054a), mean, na.rm=T)
```

```
##      Group.1          x
## 1 Männlich 13.534884
## 2 Weiblich  8.773585
```

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)

Beispieldatensatz - apply Funktion

```
ApplyDat <- cbind(1:4,runif(4),rnorm(4))
```

1	0.7621333	-1.1383110
2	0.6874500	0.0455947
3	0.4537133	0.1840527
4	0.7668416	1.4282038

Argumente der Funktion apply

?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 apply Funktion anwenden

```
apply(ApplyDat, 1, mean)
```

```
## [1] 0.2079408 0.9110149 1.2125887 2.0650151
```

```
apply(ApplyDat, 2, mean)
```

```
## [1] 2.5000000 0.6675345 0.1298851
```

Die Funktion apply

```
apply(ApplyDat, 1, var)
```

```
## [1] 1.373441 0.992411 2.414309 2.917475
```

```
apply(ApplyDat, 1, sd)
```

```
## [1] 1.1719388 0.9961983 1.5538046 1.7080617
```

```
apply(ApplyDat, 1, range)
```

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

```
## [1,] -1.138311 0.04559471 0.1840527 0.7668416
```

```
## [2,] 1.000000 2.00000000 3.0000000 4.0000000
```

```
apply(ApplyDat, 1, length)
```

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

Die Funktion tapply

?tapply

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

Beispiel Funktion tapply

```
tapply(dat$a11d054a,dat$a11d056z,mean)
```

```
##      Ambiguous answer      Item nonresponse      Not in panel 18 bis unter 20
##                      NA                      NA
##      Unit nonresponse      Not in panel 18 bis unter 20
##                      NA                      NA
## 20 bis unter 25 Jahre 25 bis unter 30 Jahre 30 bis unter 35
##                      NA                      NA
## 35 bis unter 40 Jahre 40 bis unter 45 Jahre 45 bis unter 50
##                      NA                      NA
## 50 bis unter 55 Jahre 55 bis unter 60 Jahre 60 bis unter 63
##                      NA                      NA
## 63 bis unter 65 Jahre 65 bis unter 70 Jahre    70 Jahre und mehr
##                      NA                      NA
```

```
tapply(dat$a11d054a,
       dat$a11d056z,function(x)x)
```

Links Datenanalyse

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

Grafiken und Zusammenhang

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.

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.

Datensatz

```
library(mlmRev)  
data(Chem97)
```

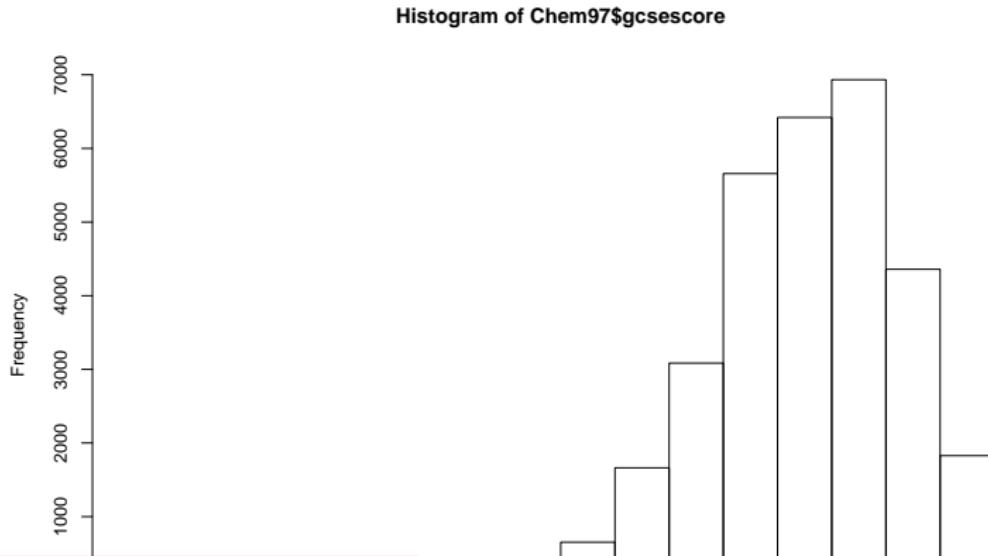
- [lea] Local Education Authority - a factor
- [school] School identifier - a factor
- [student] Student identifier - a factor
- [score] Point score on A-level Chemistry in 1997
- [gender] Student's gender
- [age] Age in month, centred at 222 months or 18.5 years
- [gcsescore] Average GCSE score of individual.
- [gcsecnt] Average GCSE score of individual, centered at mean.

Histogramm - Die Funktion hist()

Wir erstellen ein Histogramm der Variable gcsescore:

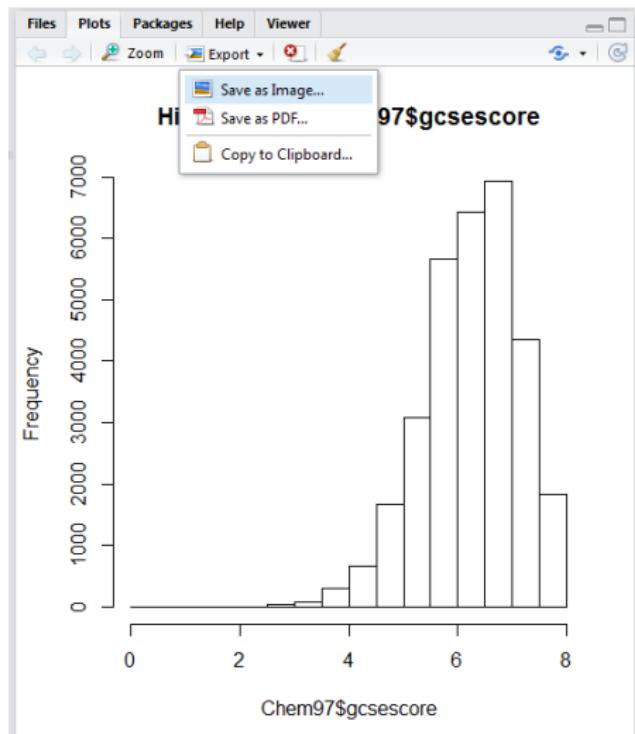
?hist

```
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()
```

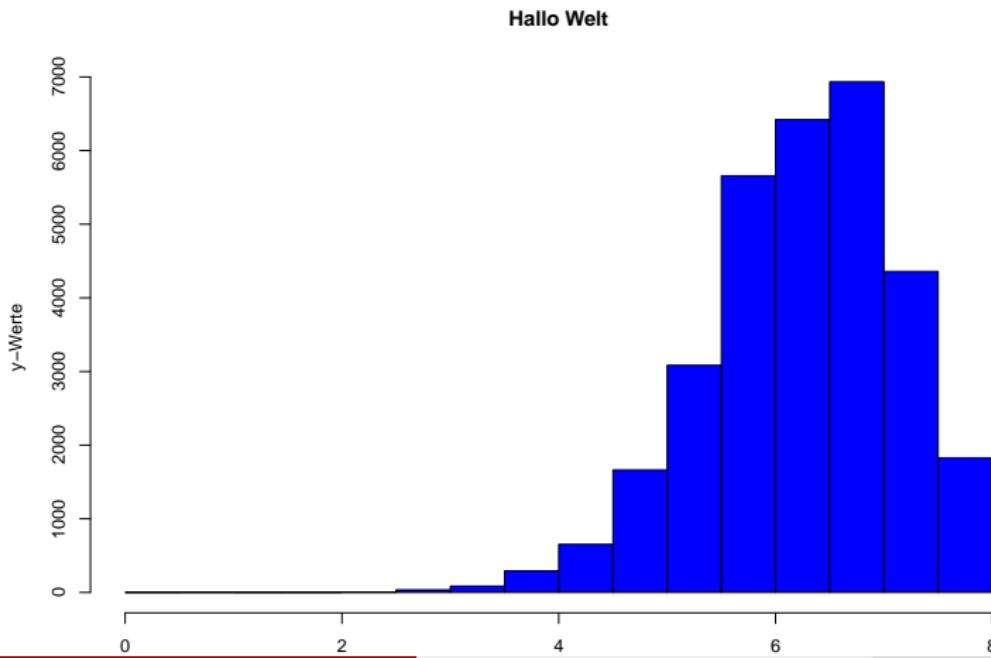
Histogramme

- 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	<code>main="Hallo Welt"</code>
xlab	x-Achsenbeschriftung	<code>xlab="x-Werte"</code>
ylab	y-Achsenbeschriftung	<code>ylab="y-Werte"</code>
col	Farbe	<code>col="blue"</code>

Histogramm

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



Barplot

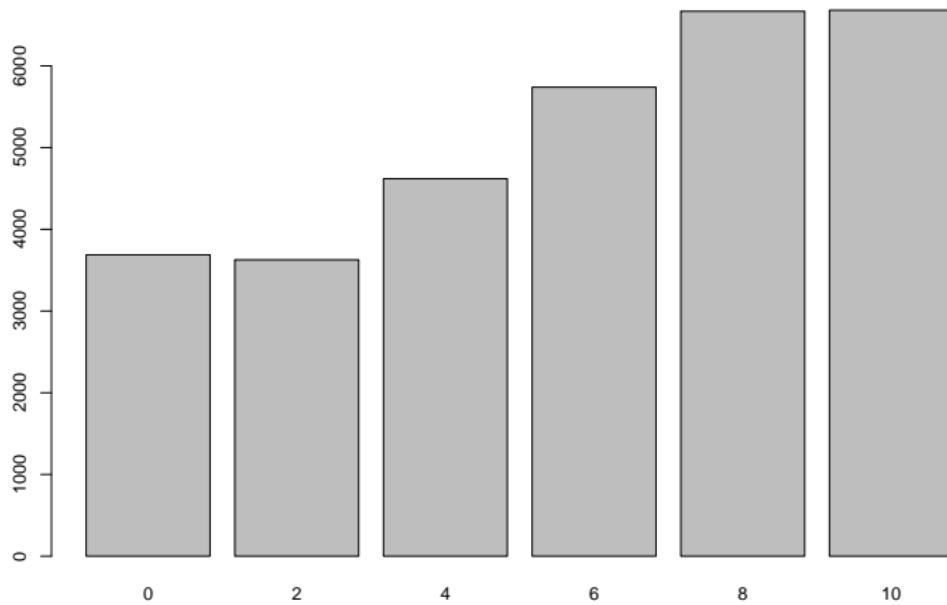
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

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

```
barplot(tabScore)
```

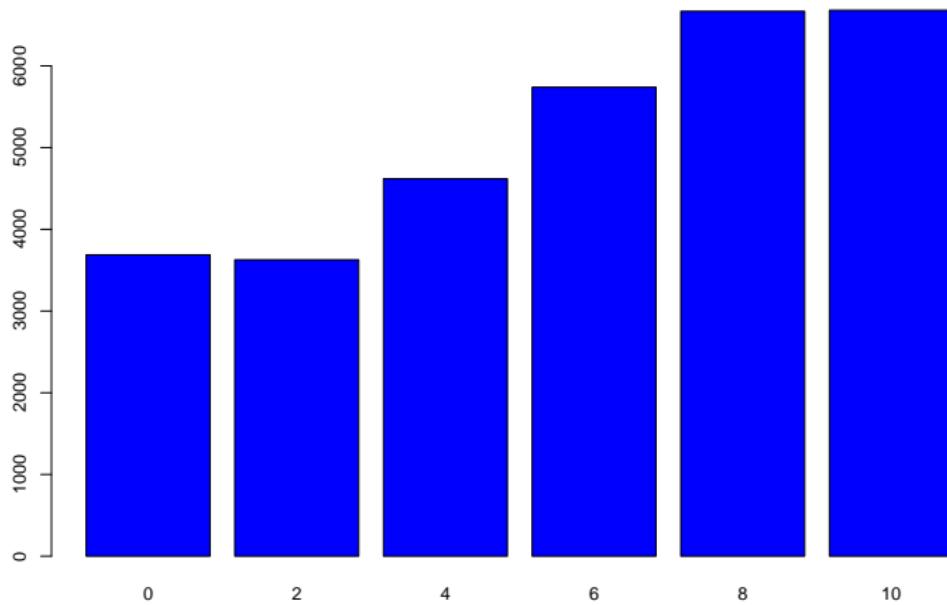
Barplots und barcharts

```
barplot(tabScore)
```



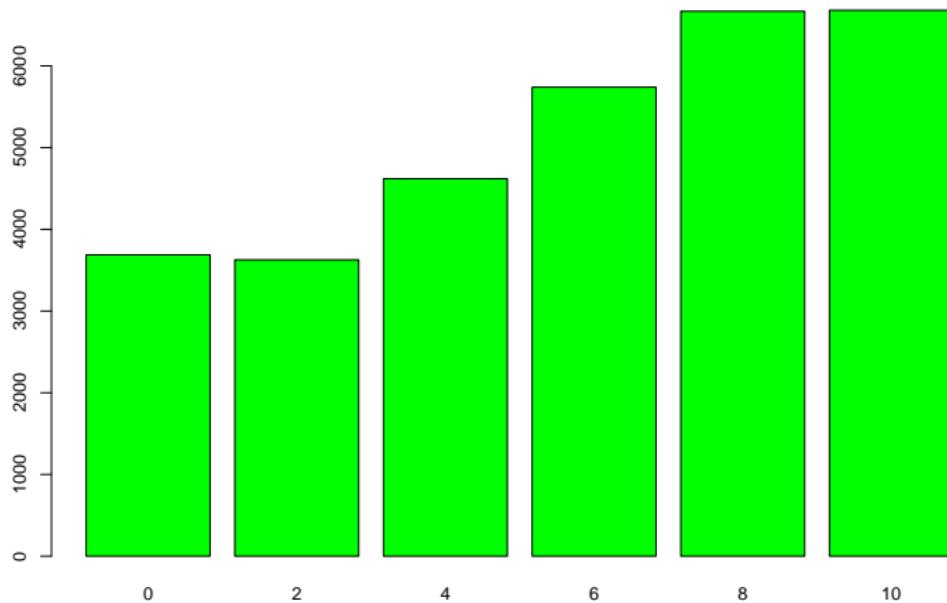
Mehr Farben:

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



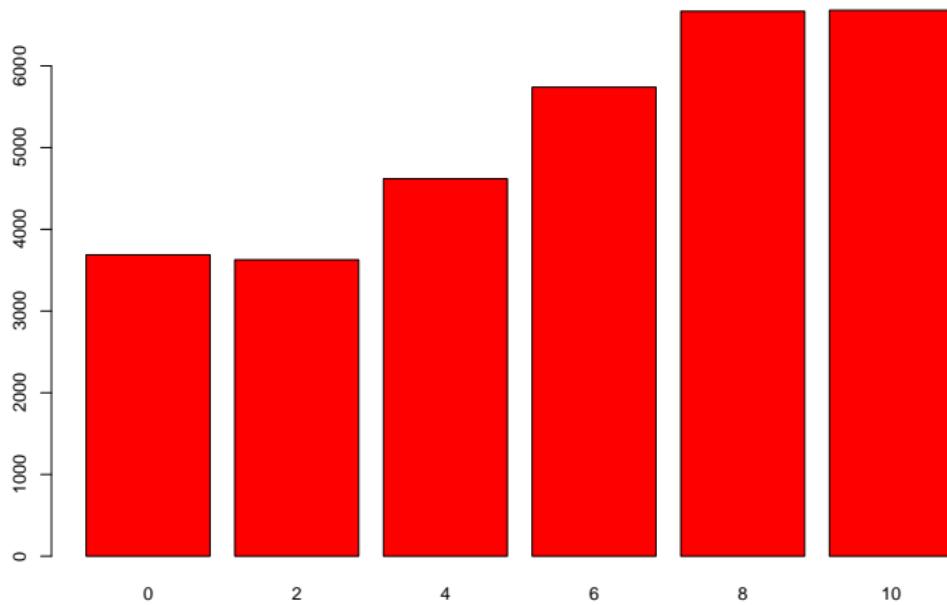
Grüne Farbe

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



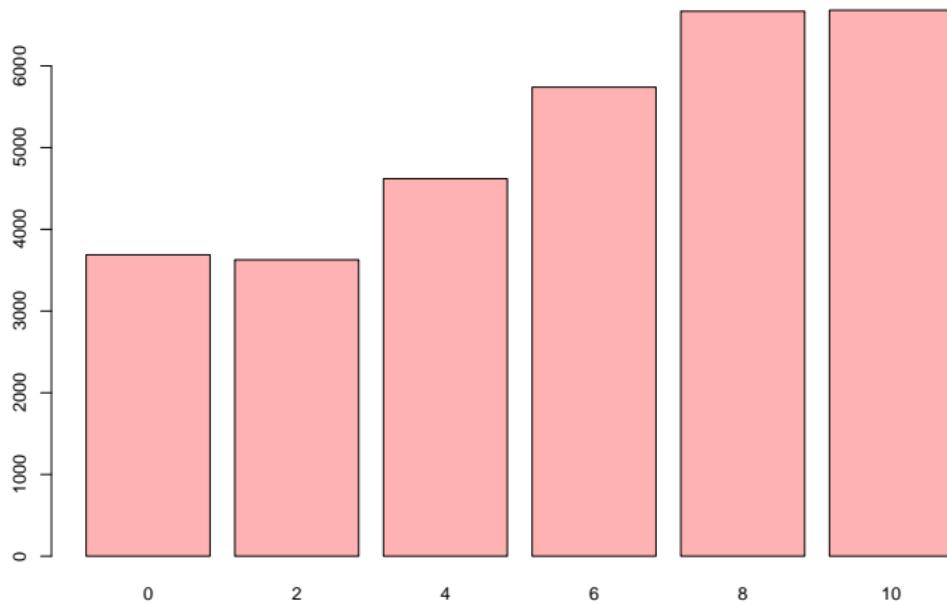
Rote Farbe

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



Transparent

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



Scatterplots

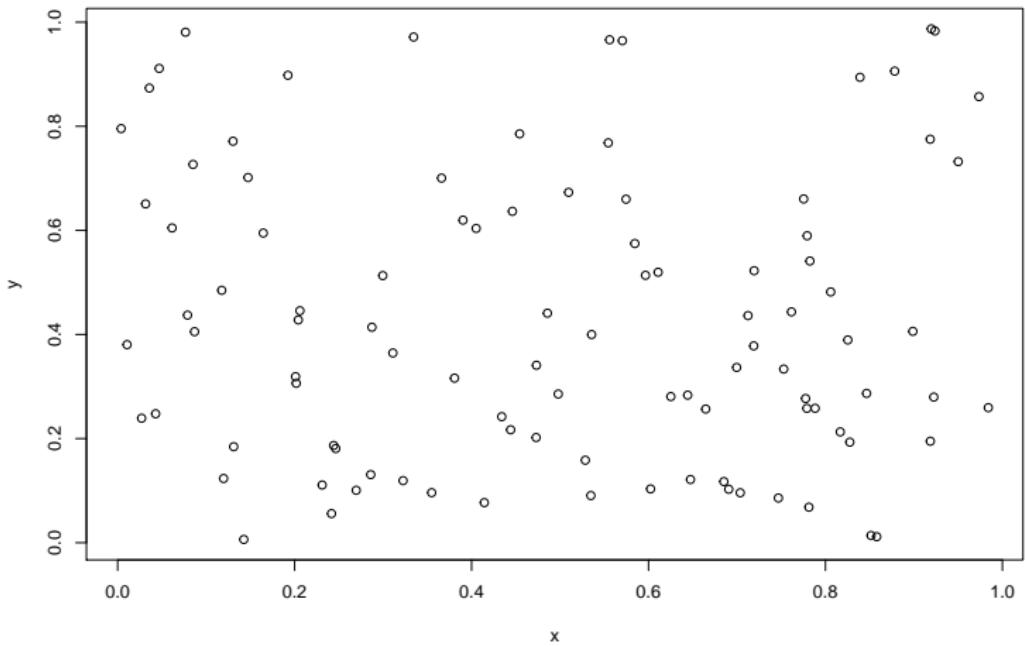
- Ein einfacher two-way Scatterplot kann mit der Funktion `plot()` erstellt werden
- `plot()` muss mindestens ein `x` und ein `y` Beobachtungsvektor übergeben werden
- Um die Farbe der Plot-Symbole anzupassen gibt es die Option `col` (Farbe als character oder numerisch)
- Die Plot-Symbole selbst können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

Beispieldaten für Scatterplot

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

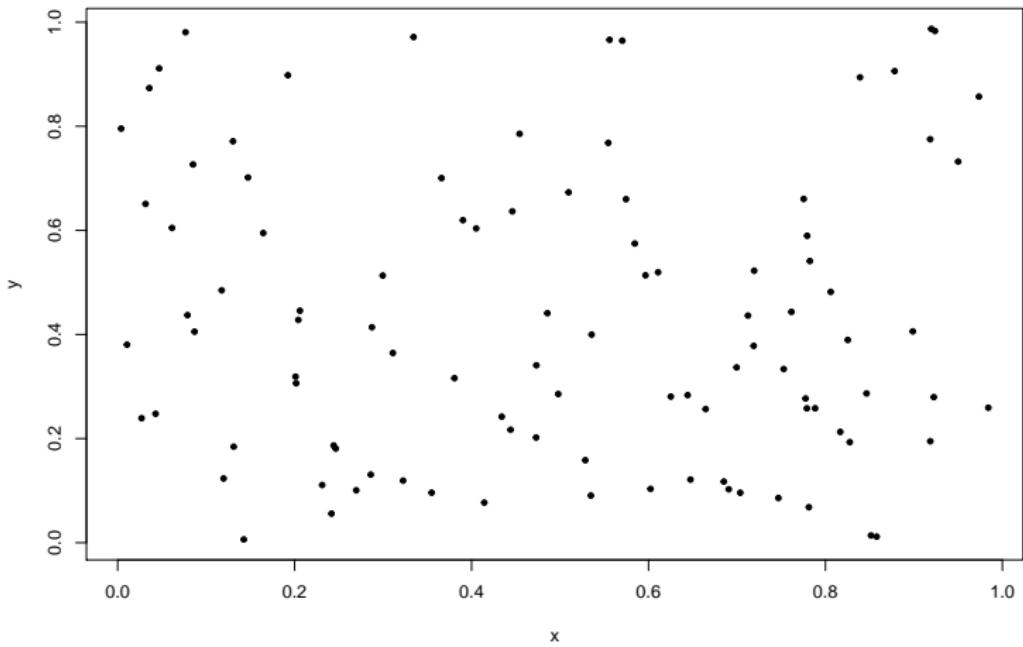
Einfacher Scatterplot

`plot(x, y)`



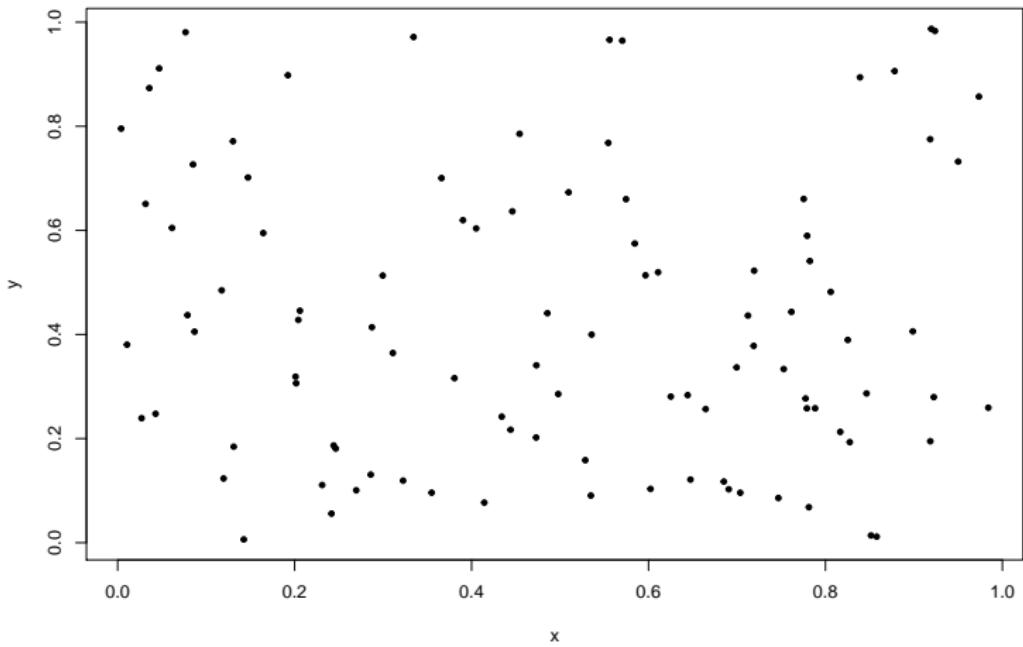
Einfacher Scatterplot II

```
plot(x,y,pch=20)
```



Einfacher Scatterplot III

```
plot(x,y,pch=20)
```



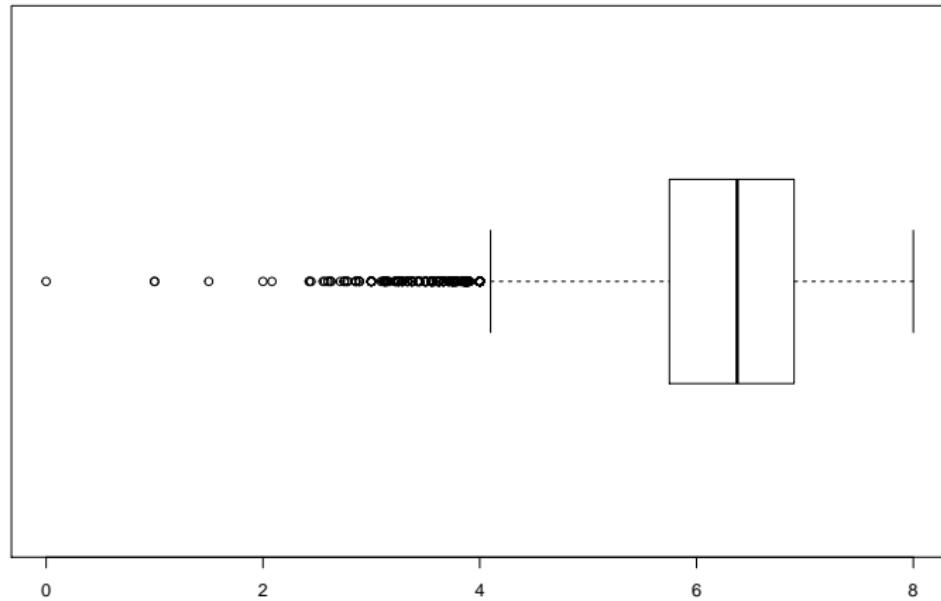
Boxplot

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

?`boxplot`

Horizontaler Boxplot

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

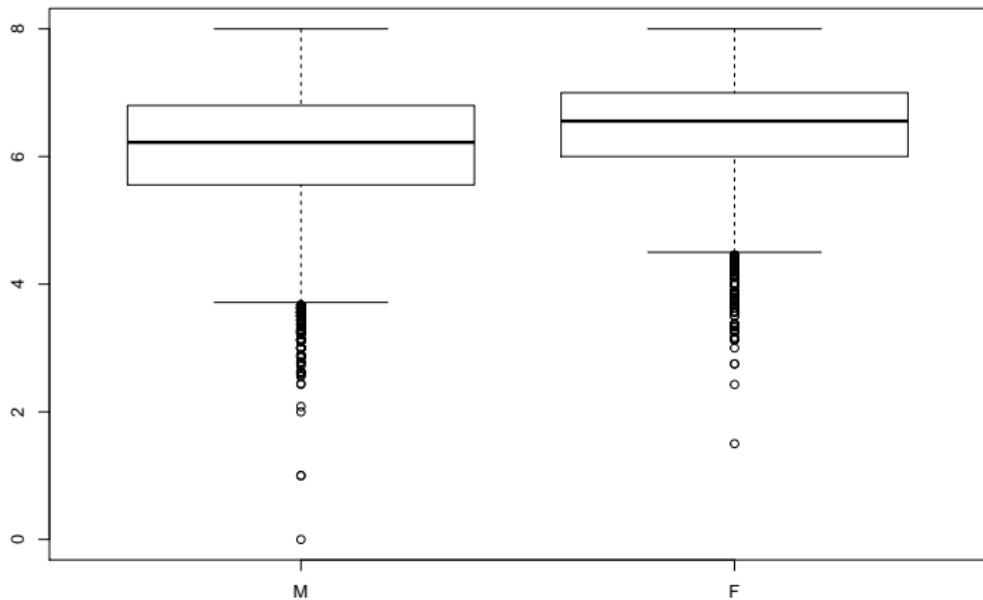


Gruppierte Boxplots

- 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 grupierter Boxplot

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



Alternativen zu Boxplot

Violinplot

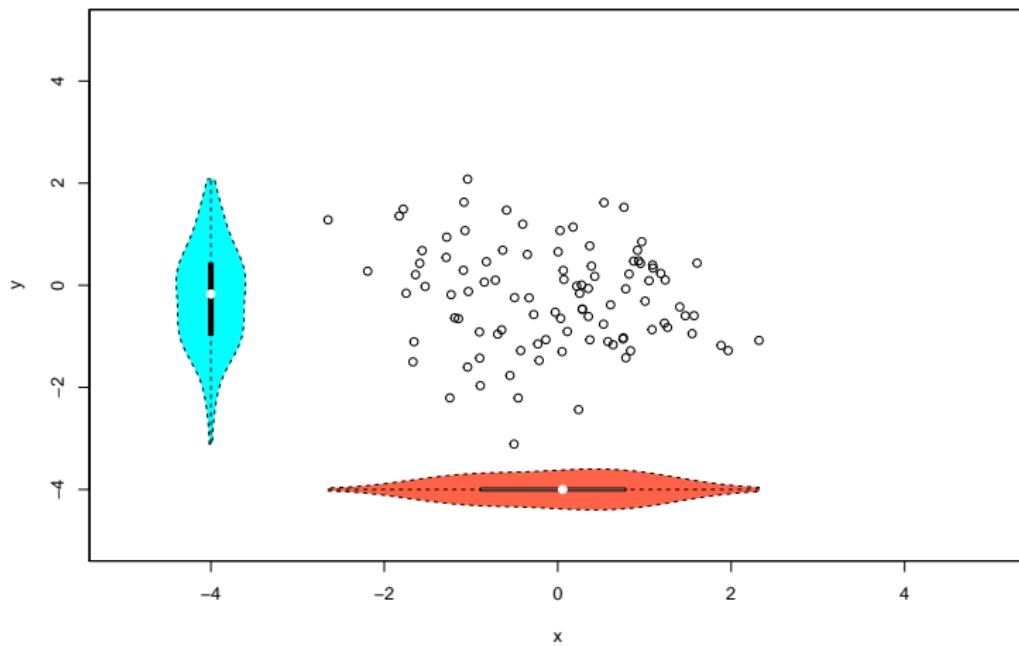
- Baut auf Boxplot auf
- Zusätzlich Informationen über Dichte der Daten
- Dichte wird über Kernel Methode berechnet.
- weißer Punkt - Median
- Je weiter die Ausdehnung, desto größer ist die Dichte an dieser Stelle.

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

Die Bibliothek vioplot

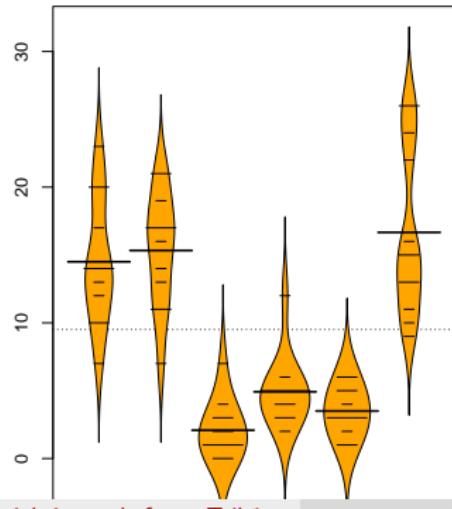
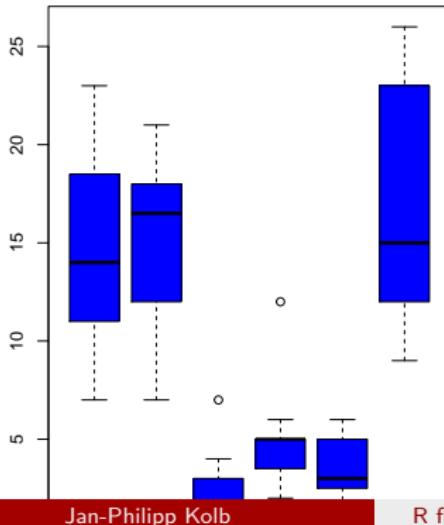
```
library(vioplot)
plot(x, y, xlim=c(-5,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-4,
        add=TRUE, lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=-4,
        add=TRUE, lty=2)
```

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")
```

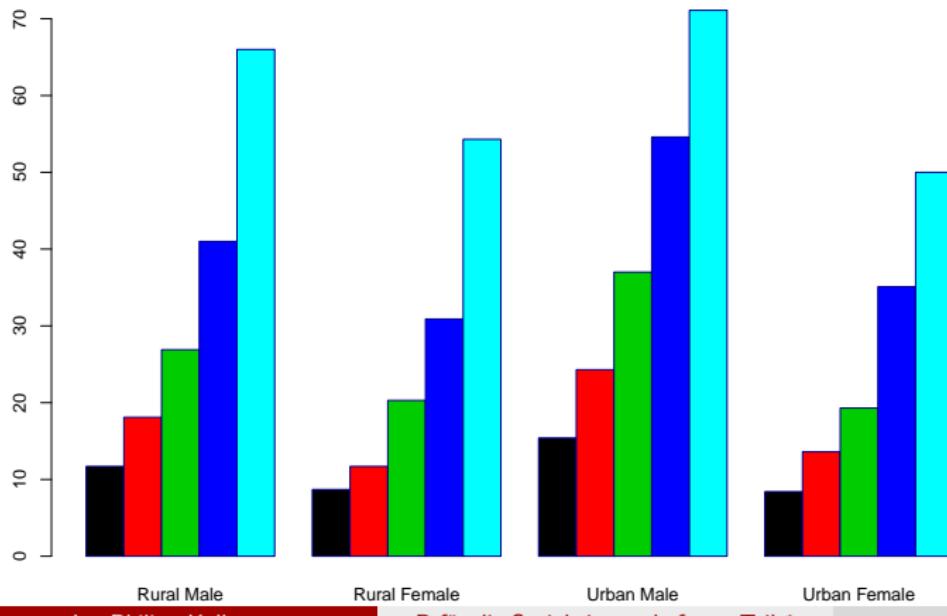


CMYK Farbschema

```
pdf("test.cmyk.pdf", colormodel='cmyk')
pie(1:10, col=1:10)
dev.off()
```

Aufgabe - einfache Grafiken

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



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](http://stat.ethz.ch/R-manual/R-devel/library/lattice/html/Lattice.html)

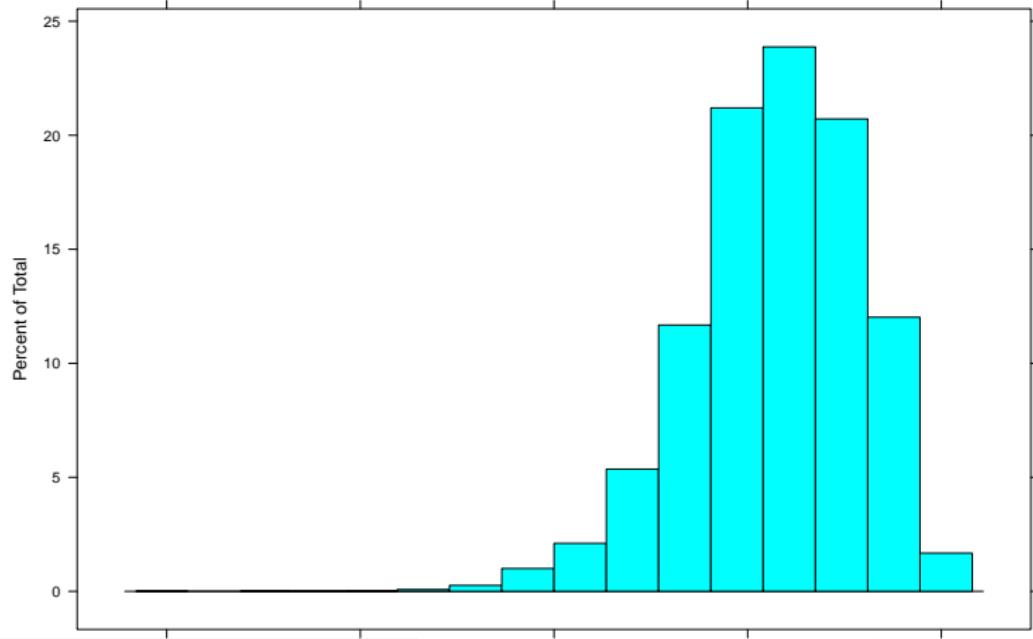
Der Datensatz - Scores on A-level Chemistry in 1997

```
library("mlmRev")
data(Chem97)
```

variables	categories
lea	Local Education Authority
school	School identifier
student	Student identifier
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

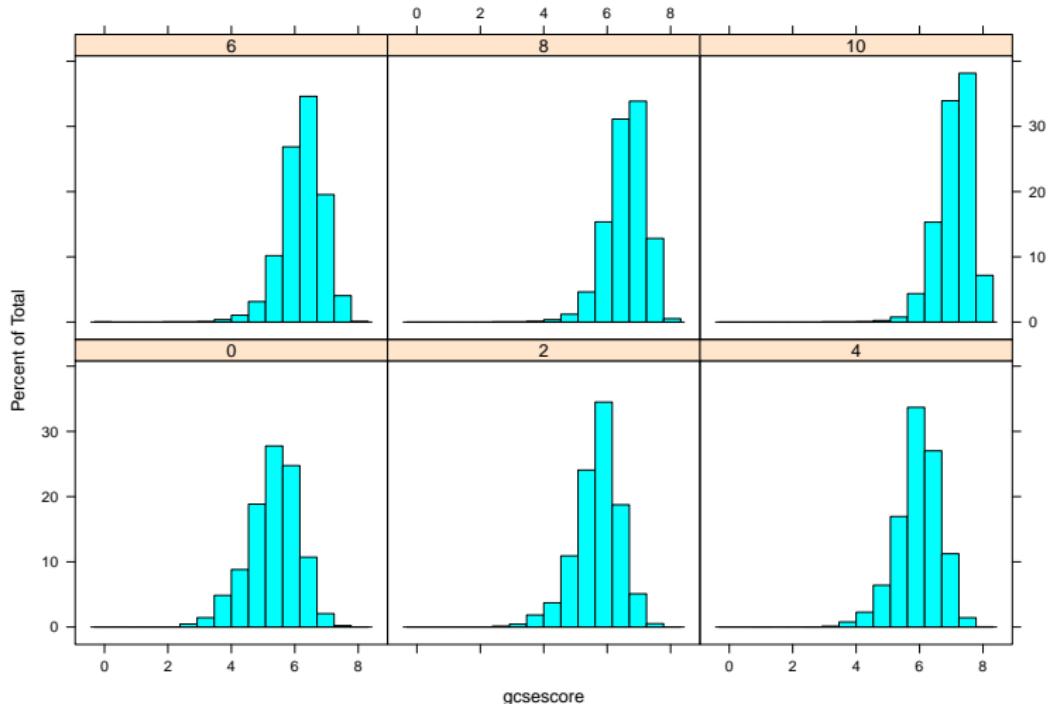
Histogramm mit Lattice

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



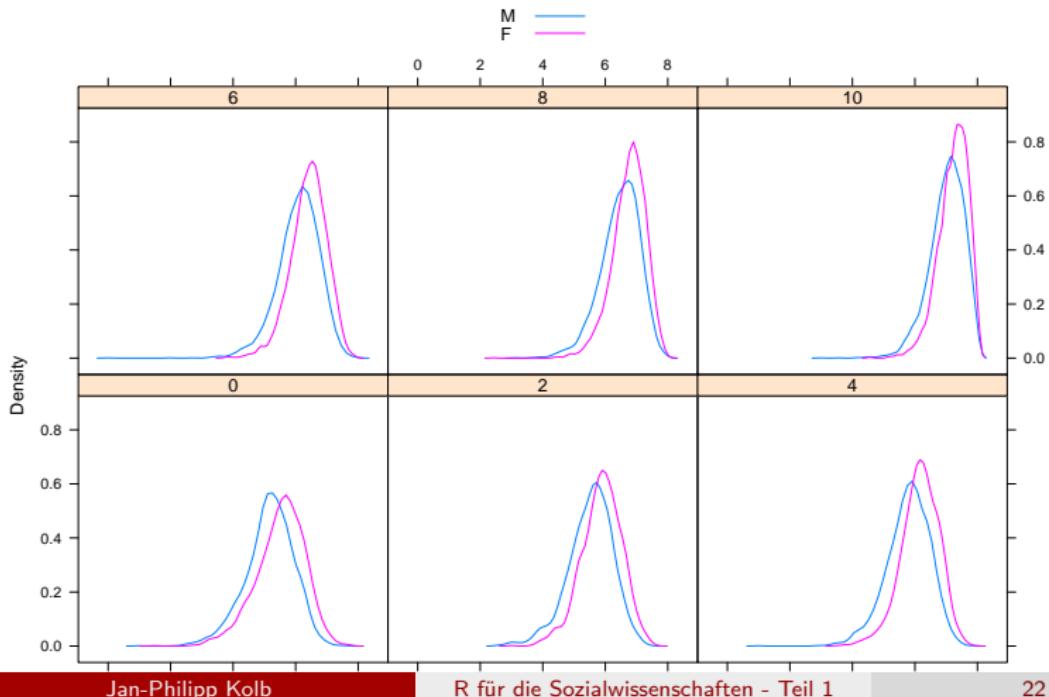
Histogramm mit Lattice

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



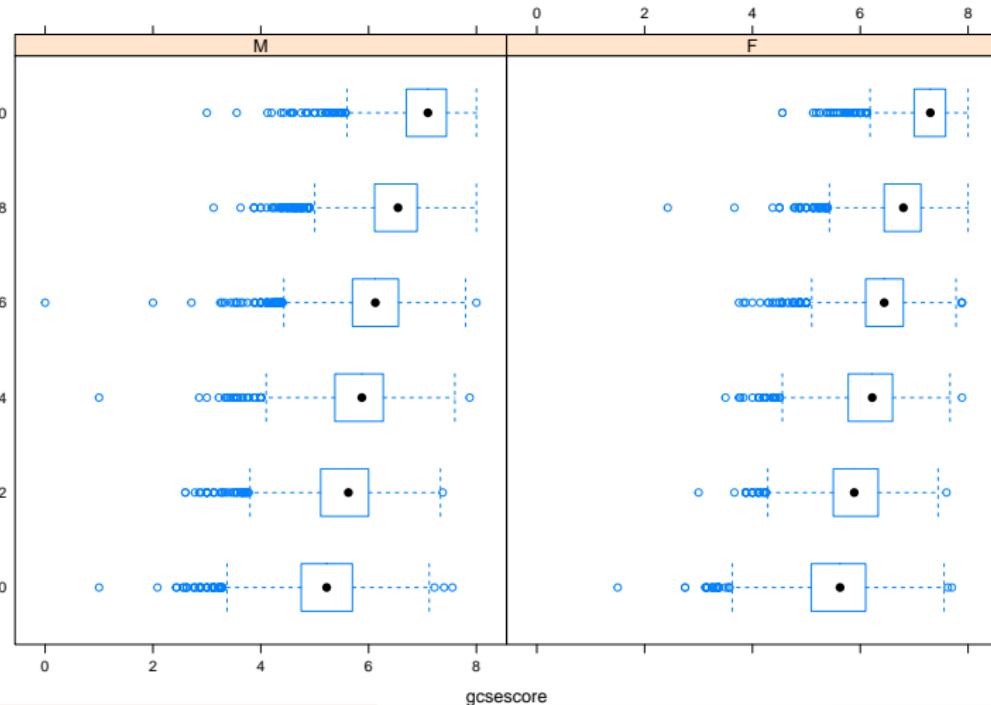
Die Dichte mit Lattice zeichnen

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



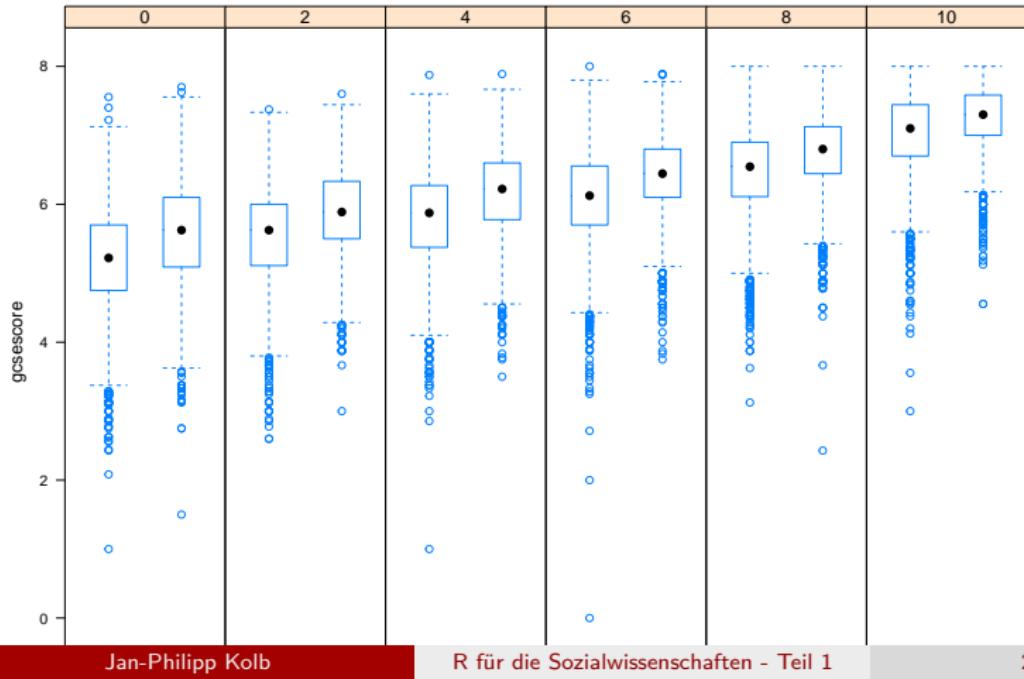
Boxplot mit Lattice zeichnen

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



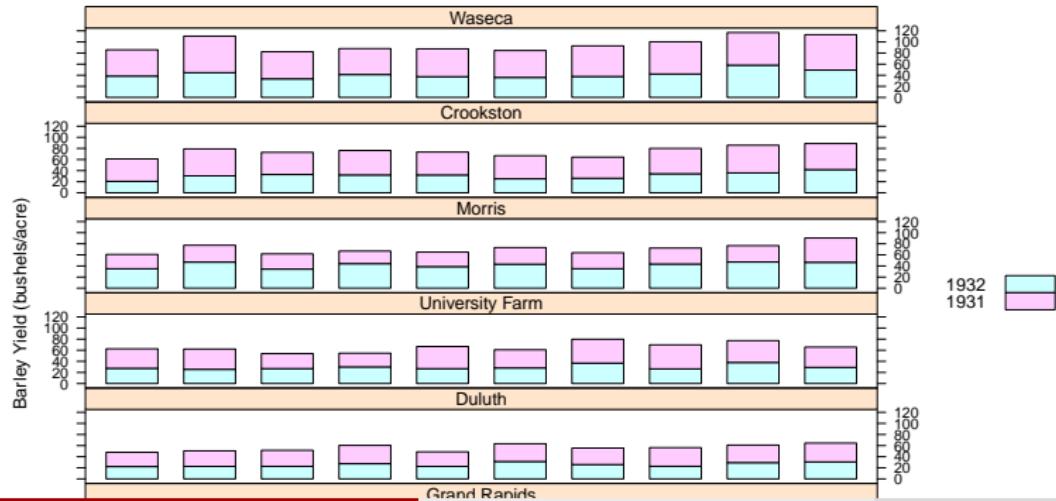
Boxplot mit Lattice zeichnen

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



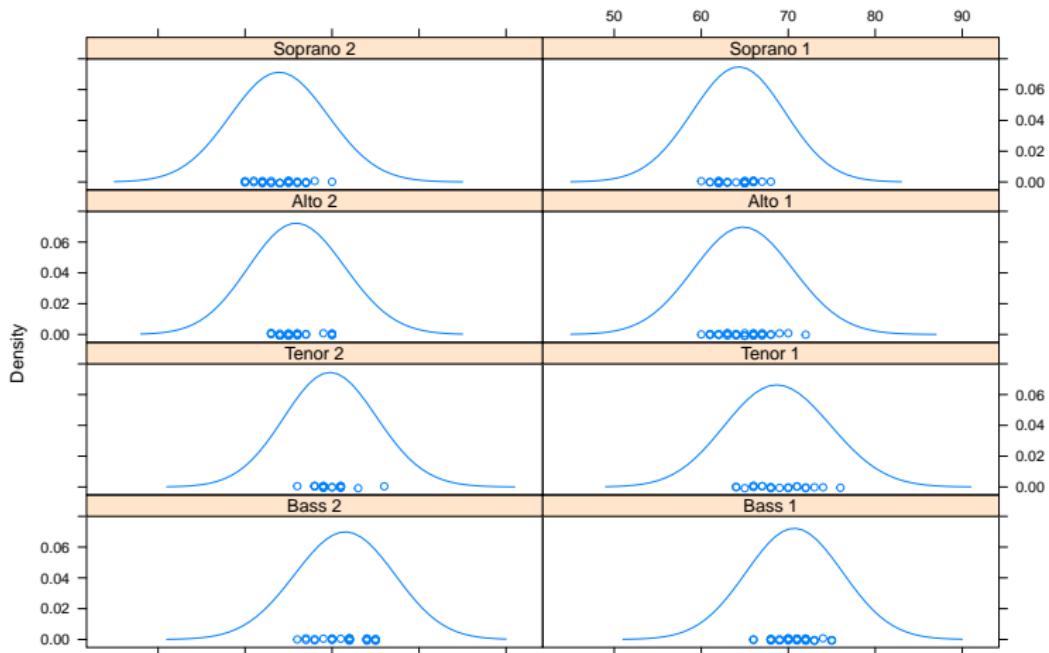
Univariate Plots

```
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)))
```



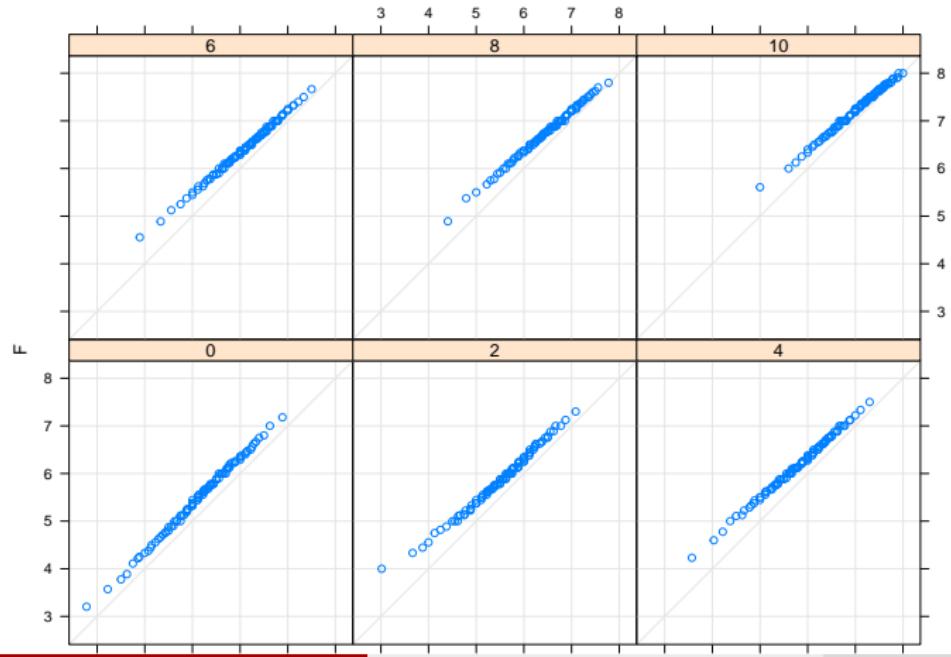
Densityplot

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



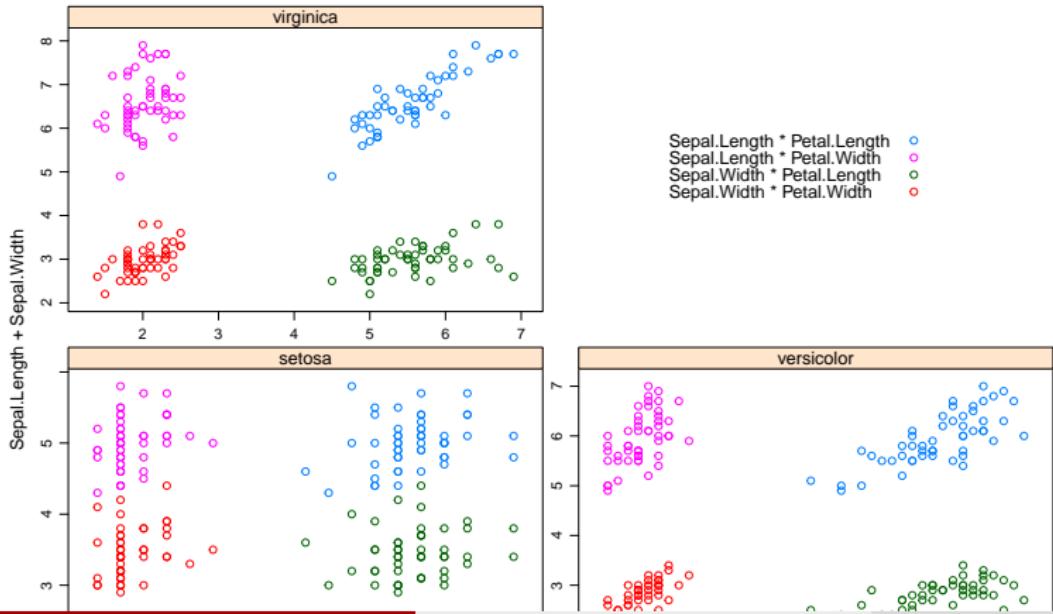
Bivariate Plots

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



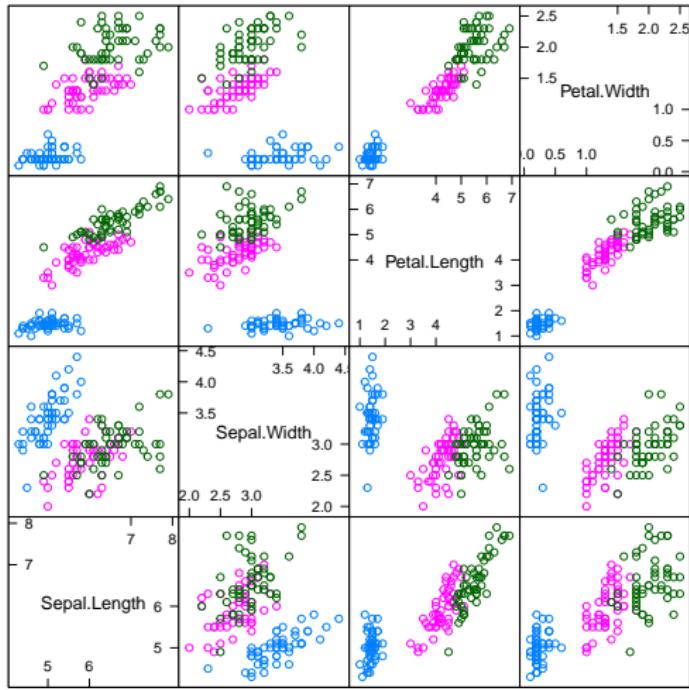
xyplot

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



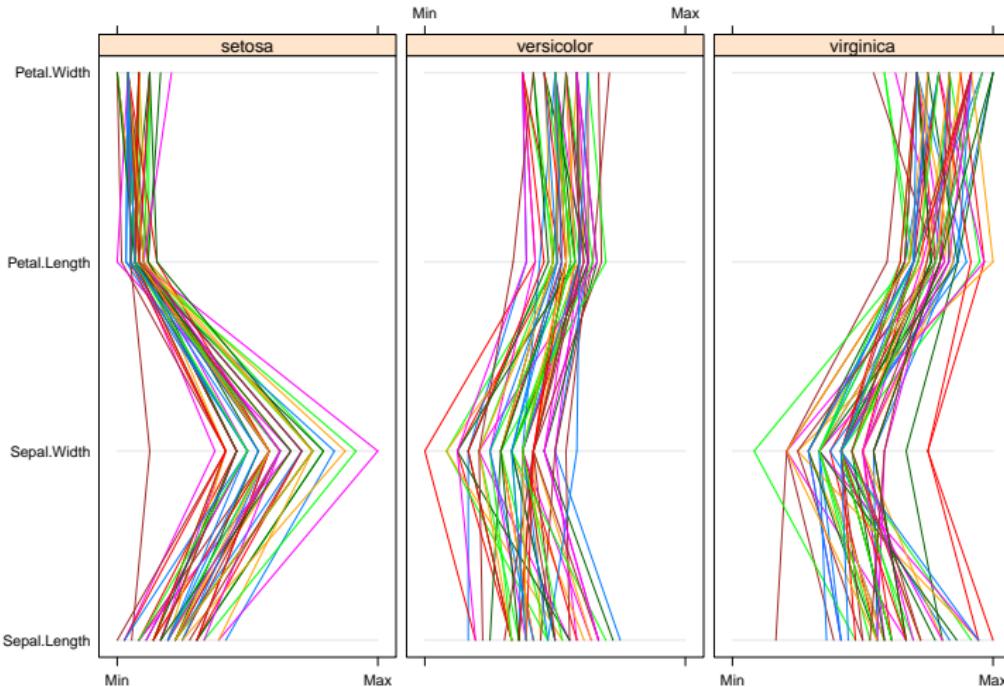
Multivariate Plots

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



parallelplot

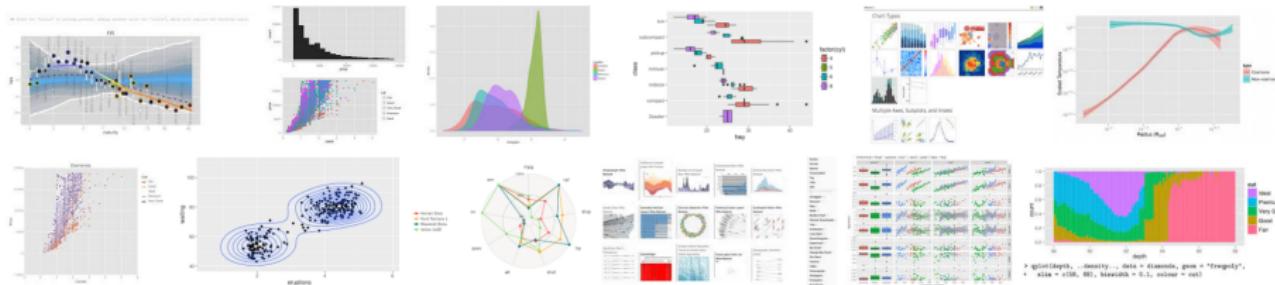
```
parallelplot(~iris[1:4] | Species, iris)
```



ggplot und ggmap

Das Paket ggplot2

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



Das Paket `ggplot2` installieren und laden

- Basiseinführung `ggplot2`

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

```
library(ggplot2)
```

Der diamonds Datensatz

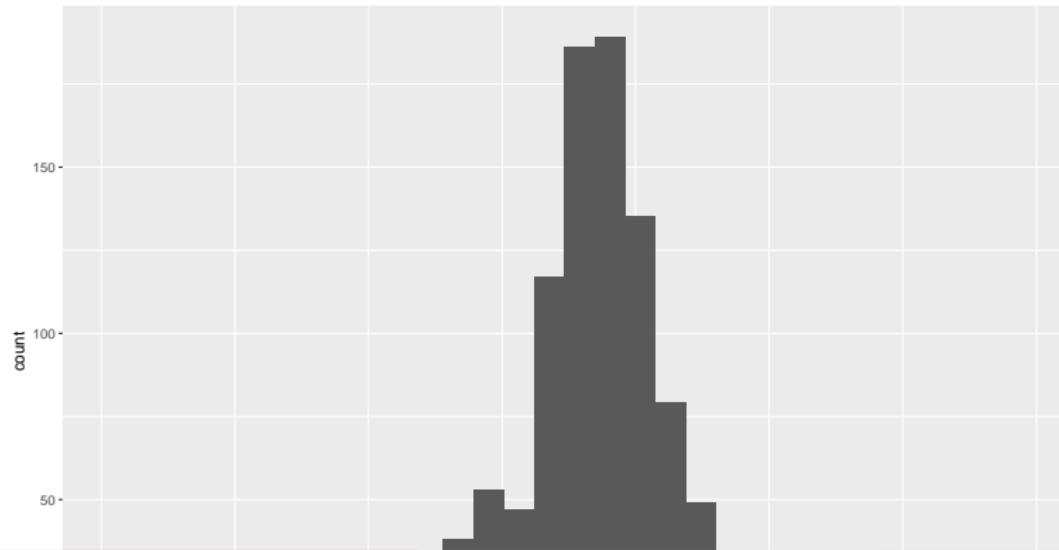
```
head(diamonds)
```

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
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

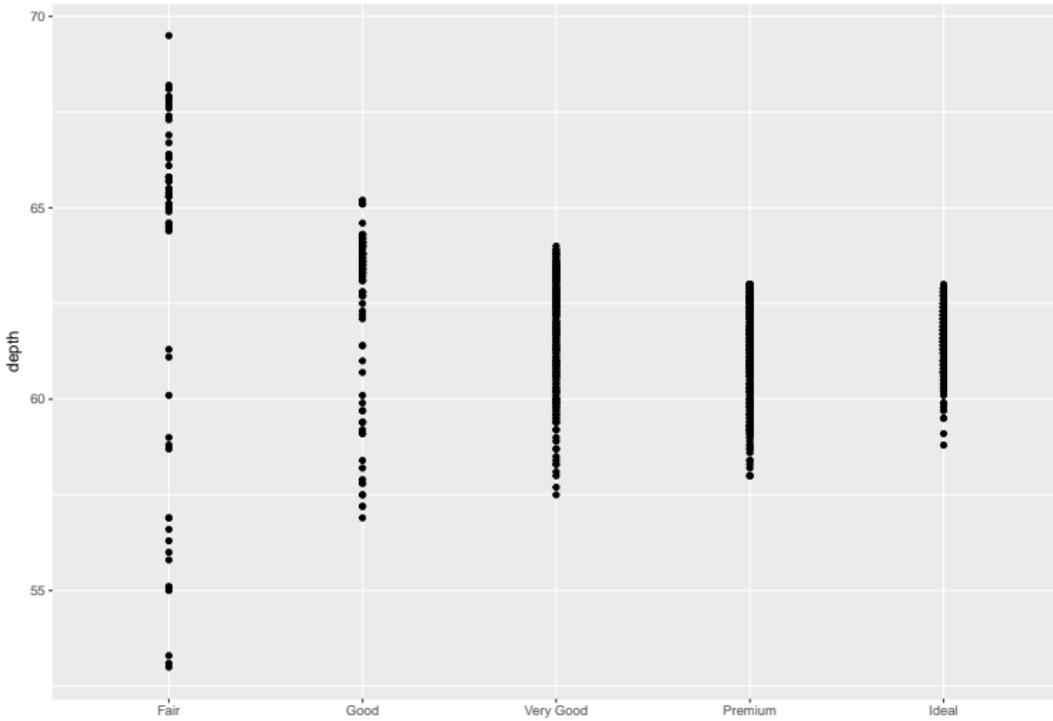
- qplot wird für schnelle Graphiken verwendet (quick plots)
- bei ggplot kann man alles bis ins Detail kontrollieren

```
# histogram  
qplot(depth, data=diamonds2)
```



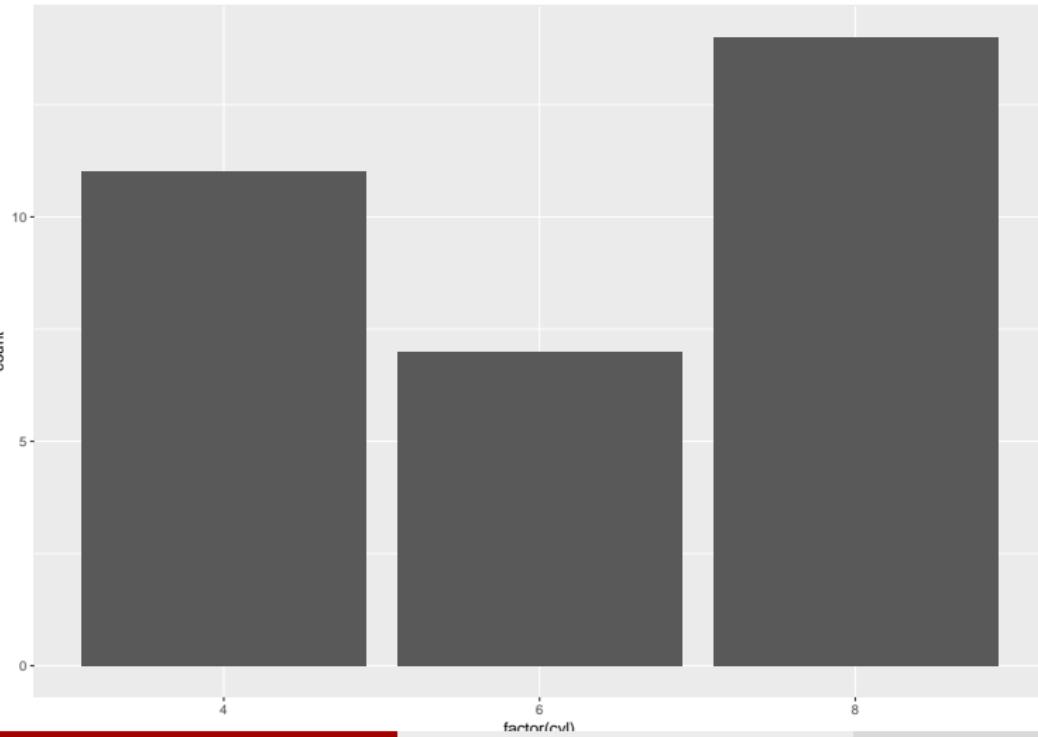
Ein Balkendiagramm

```
qplot(cut, depth, data=diamonds2)
```



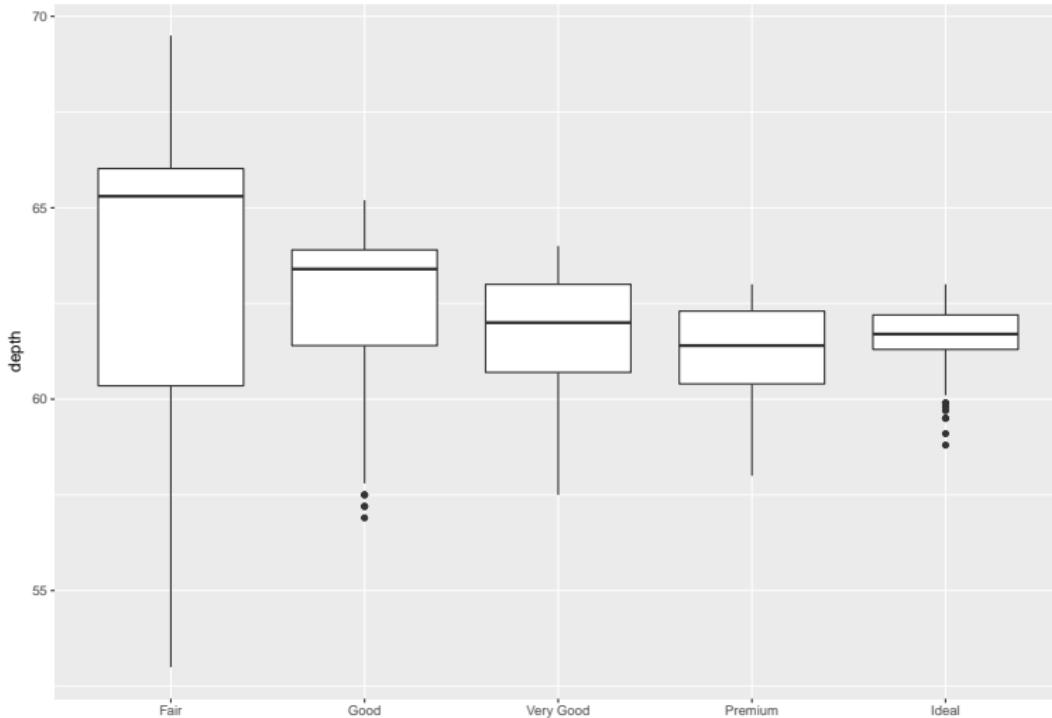
Ein weiteres Balkendiagramm

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



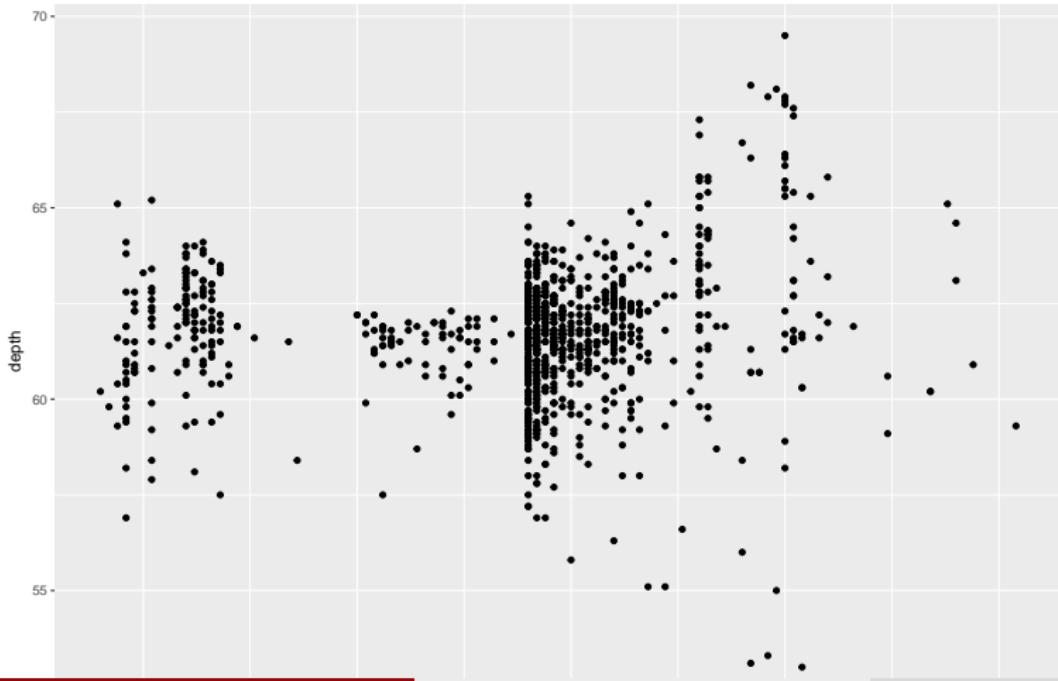
Boxplot

```
qplot(data=diamonds2, x=cut, y=depth, geom="boxplot")
```



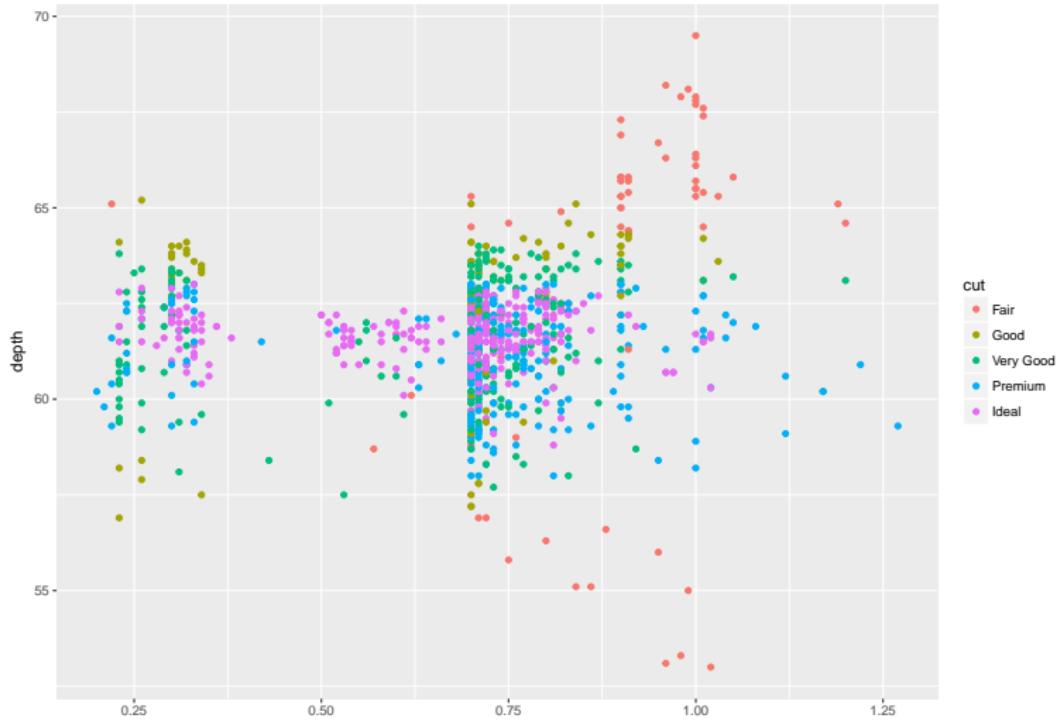
Scatterplot

```
# scatterplot  
qplot(carat, depth, data=diamonds2)
```



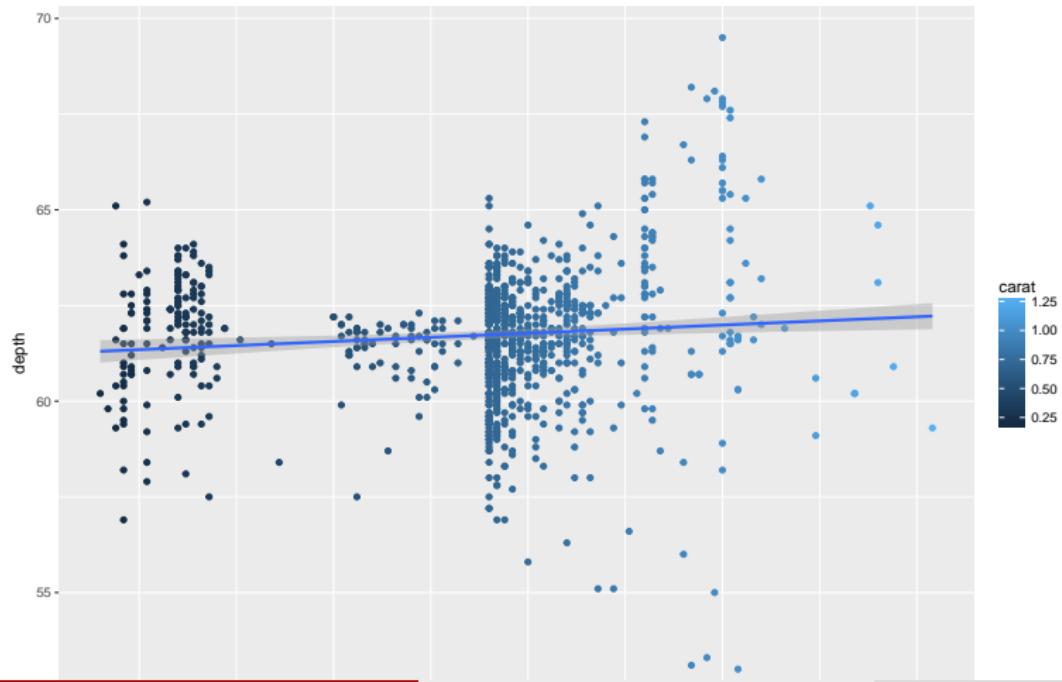
Farbe hinzufügen:

```
qplot(carat, depth, data=diamonds2, color=cut)
```



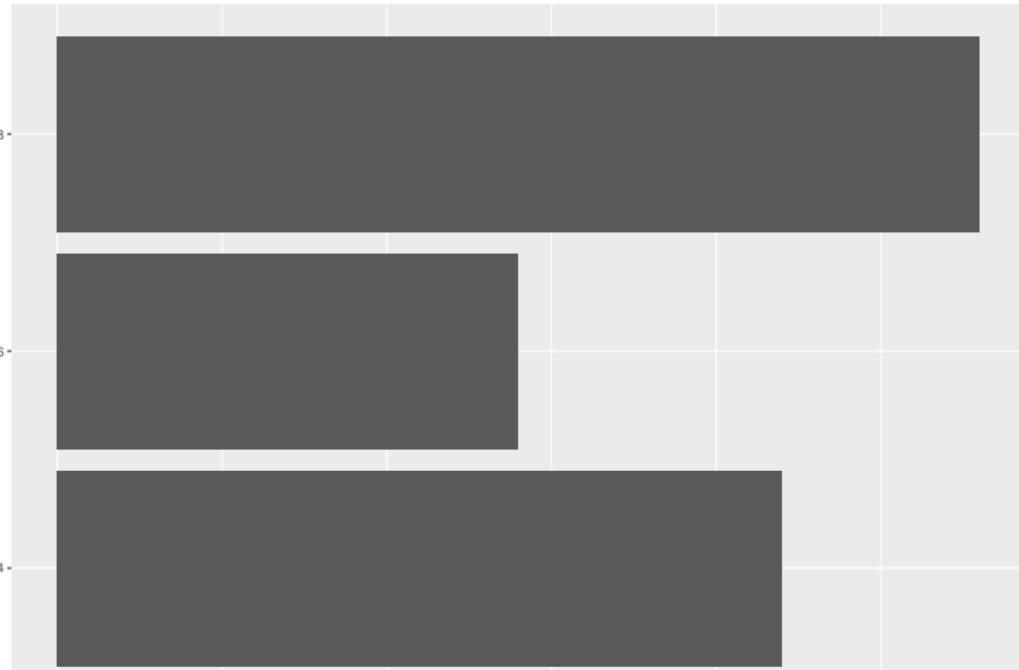
Trendlinie hinzufügen

```
myGG<-qplot(data=diamonds2, x=carat, y=depth, color=carat)  
myGG + stat_smooth(method="lm")
```



Graphik drehen

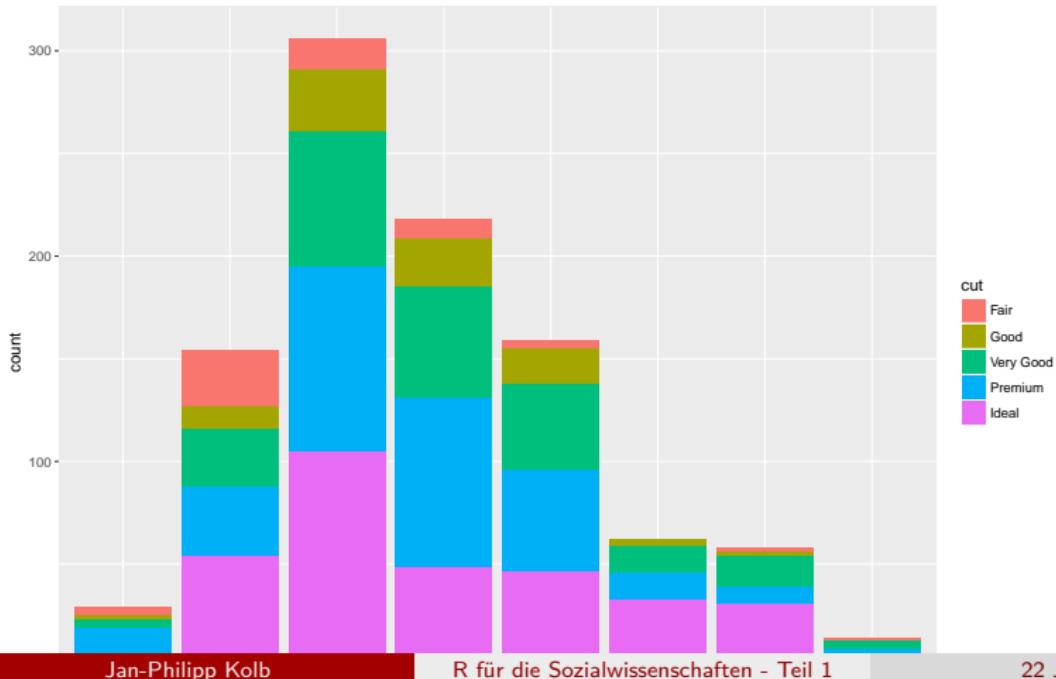
```
qplot(factor(cyl), data=mtcars, geom="bar") +  
coord_flip()
```



Wie nutzt man ggplot

- die aesthetics:

```
ggplot(diamonds2, aes(clarity, fill=cut)) + geom_bar()
```



Farben selber wählen

Es wird das Paket RColorBrewer verwendet um die Farbpalette zu ändern

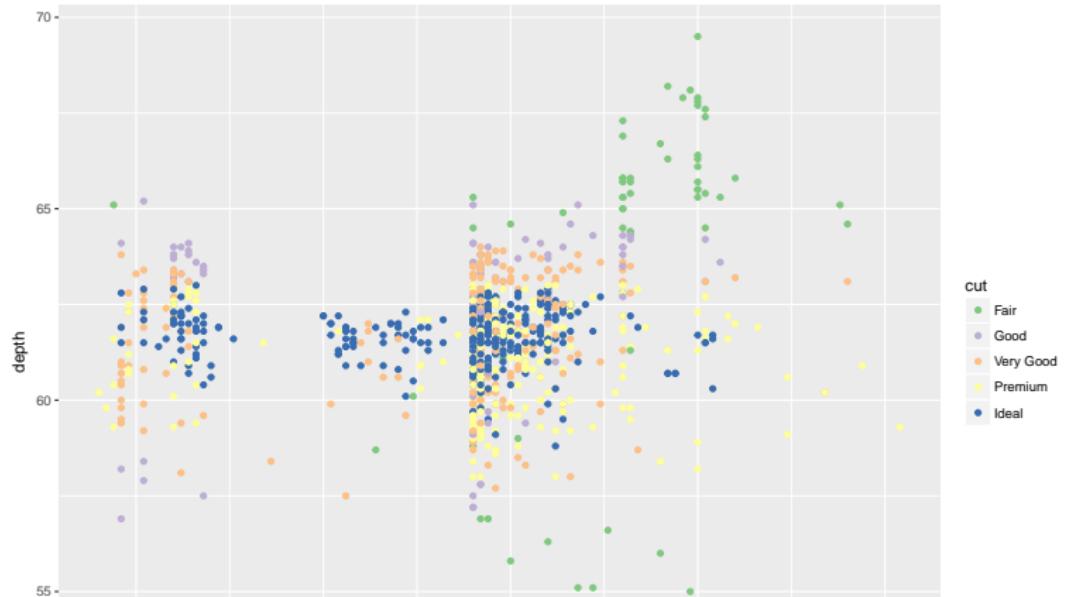
```
install.packages("RColorBrewer")

library(RColorBrewer)
myColors <- brewer.pal(5, "Accent")
names(myColors) <- levels(diamonds2$cut)
colScale <- scale_colour_manual(name = "cut",
                                  values = myColors)
```

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

Eine Graphik mit den gewählten Farben

```
p <- ggplot(diamonds2,aes(carat, depth, colour = cut)) +  
  geom_point()  
p + colScale
```



Speichern mit ggsave

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

Links

- Warum man ggplot2 für einfache Grafiken nutzen sollte

Why I use ggplot2

February 12, 2016

By David Robinson

 Like 585

 Share

 Share 69

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

590
SHARES

 Share

 Tweet

- Einführung in ggplot2

Installieren des Paketes

- Zur Erstellung der Karten brauchen wir das Paket `ggmap`:

```
install.packages("ggmap")
```

Paket ggmap - Hallo Welt

```
library(ggmap)
```

Und schon kann die erste Karte erstellt werden:

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



Zoom level bei ggmap

- 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

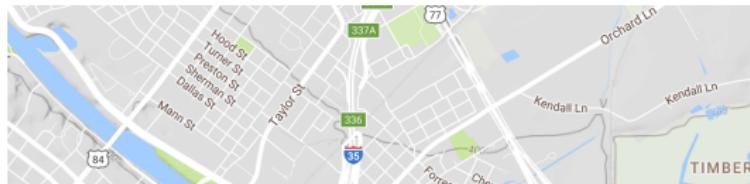
Ausschnitt aus der Hilfe Seite zum Befehl qmap:

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")
```

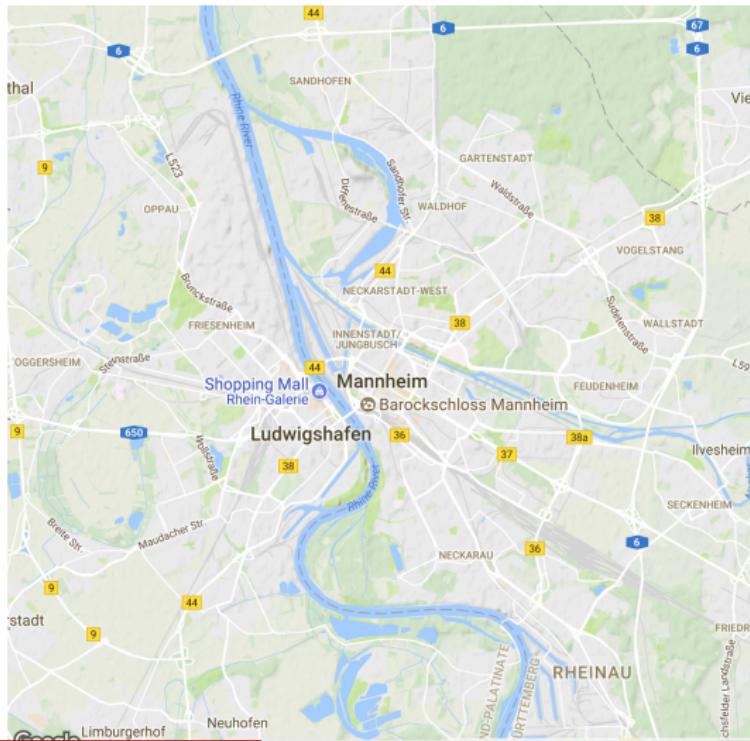
Das Beispiel kann man direkt in die Konsole kopieren:

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



Ein anderes *zoom level*

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



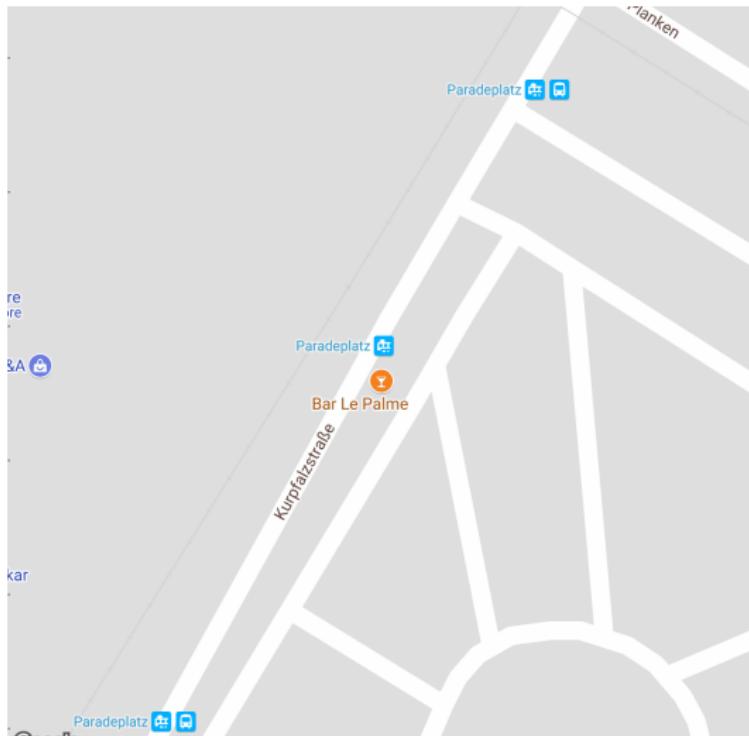
Näher rankommen

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



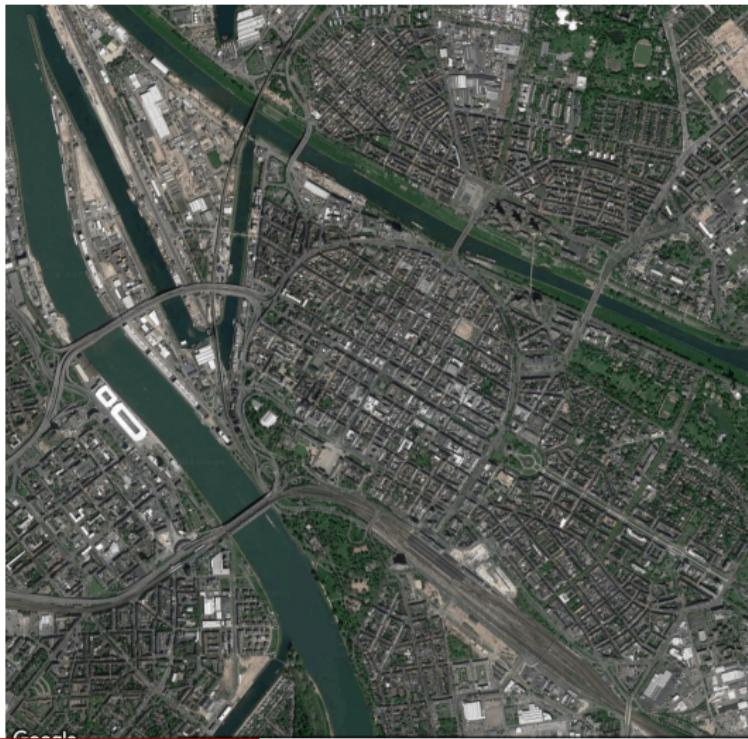
Ganz nah dran

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



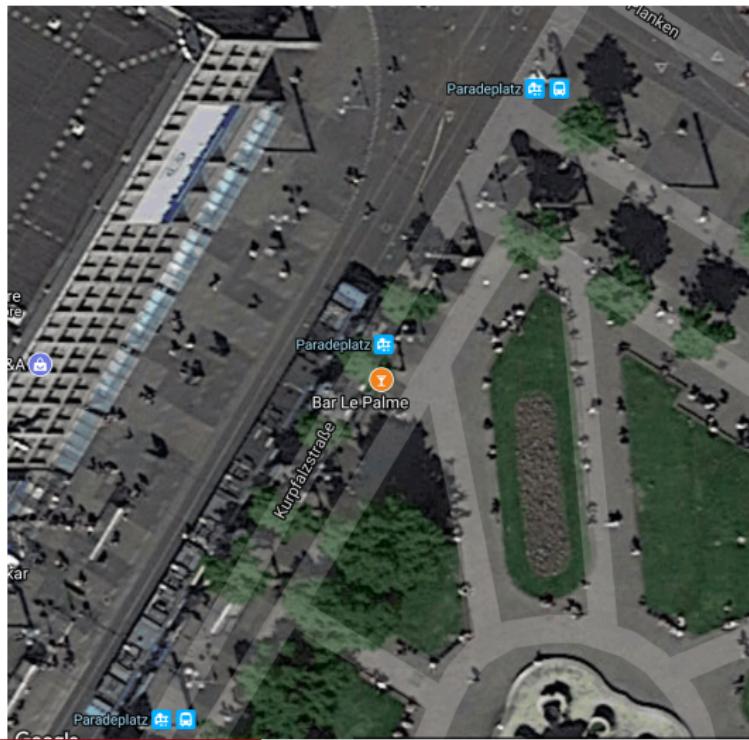
ggmap - maptype satellite

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



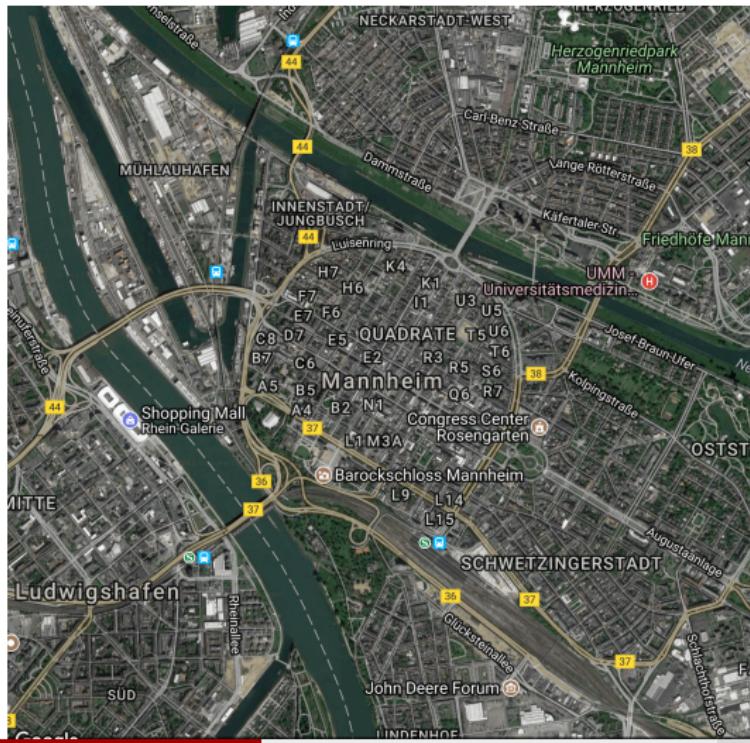
ggmap - maptype satellite zoom 20

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



ggmap - maptype hybrid

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

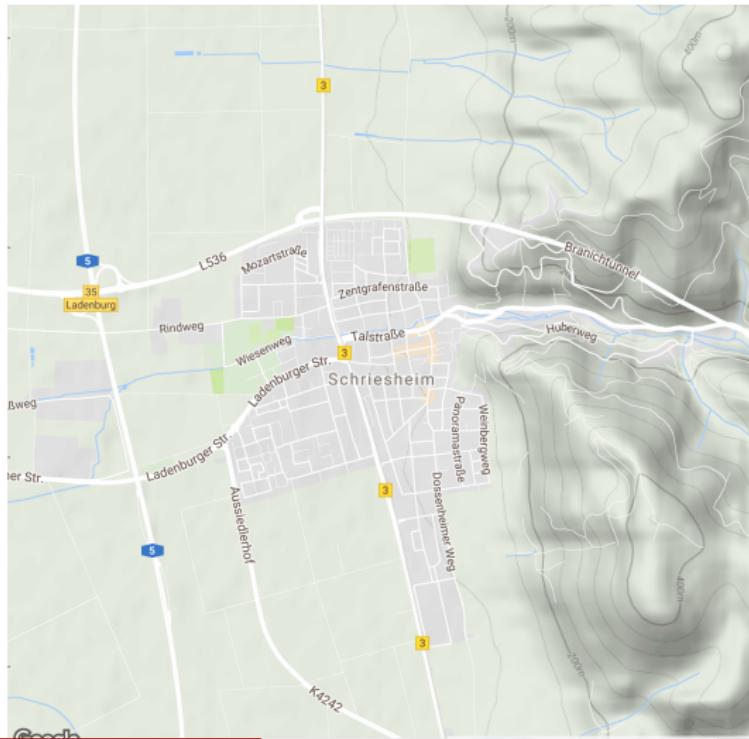


Terrain/physical maps

- 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)



- Abstraktion wird genutzt um nur essentielle Informationen zu zeigen.
- Bsp. U-Bahn Karten - wichtig sind Richtungen und wenig Infos zur Orientierung
- Nun kommen Karten, die sich als Hintergrund 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")
```

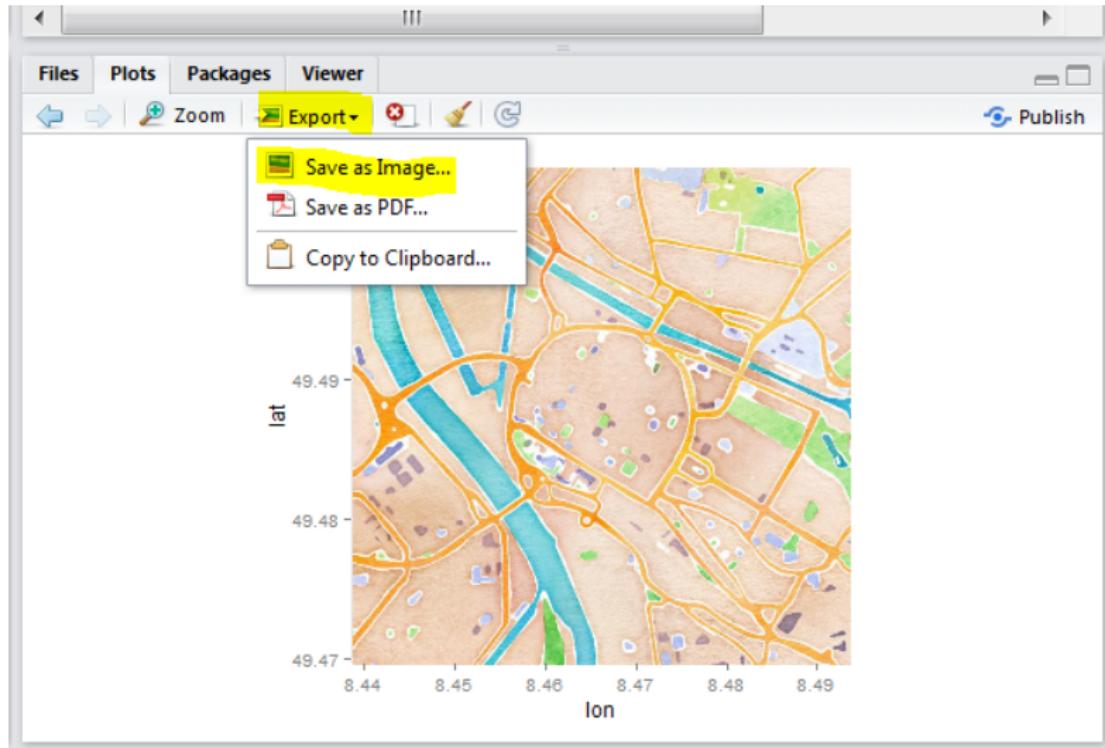


ggmap - maptype terrain-lines

```
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 . . .

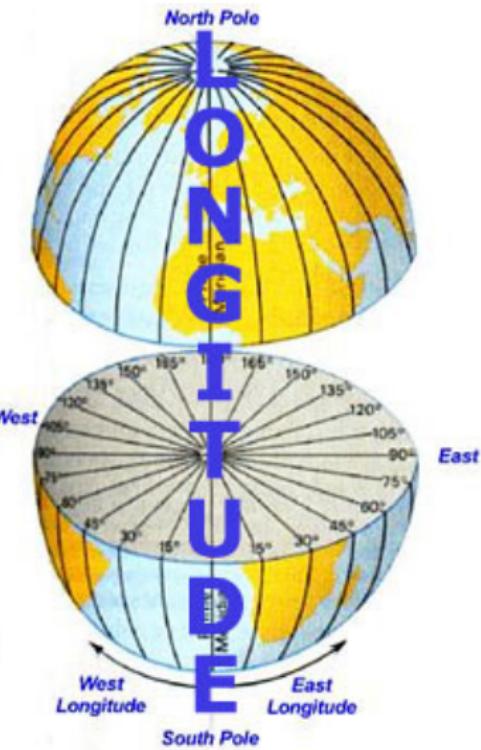
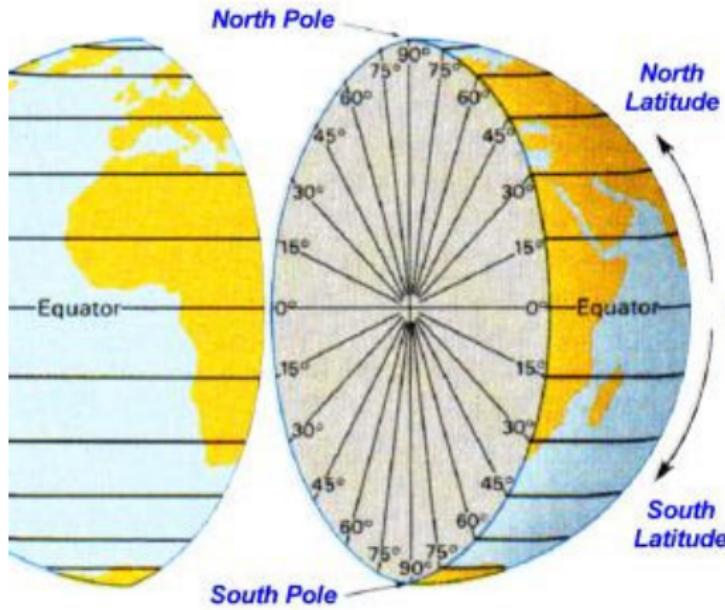
Wikipedia - Geocoding

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

lon	lat
8.463182	49.48608

Latitude und Longitude

LATITUDE



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      miles seconds
## 1 Q1, 4 Mannheim B2, 1 Mannheim 746 0.746 0.4635644      209
##                 hours
## 1 0.05805556
```

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

```
##                 from                  to      m      km      miles seconds
## 1 Q1, 4 Mannheim B2, 1 Mannheim 546 0.546 0.3392844      423
```

Eine andere Distanz bekommen

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim",mode="bicycling")  
  
##           from          to    m      km    miles seconds  
## 1 Q1, 4 Mannheim B2, 1 Mannheim 555 0.555 0.344877     215  
##           hours  
## 1 0.05972222
```

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 8.475754 49.48304
```

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

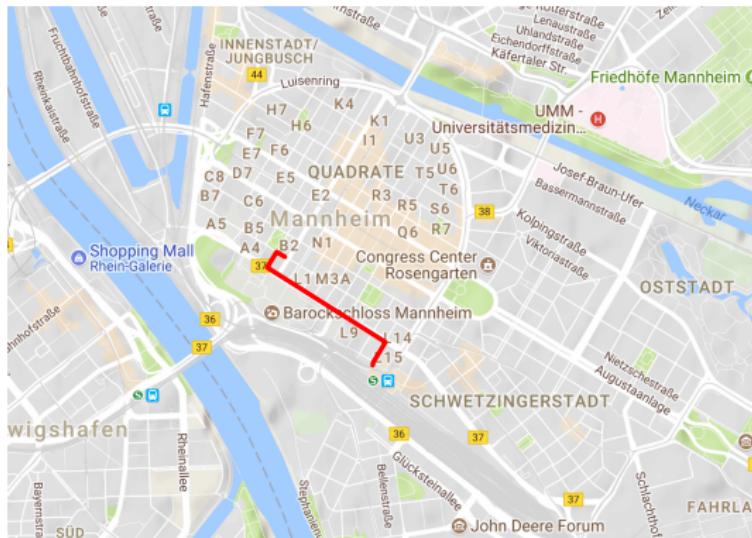
```
from <- "Mannheim Hbf"  
to <- "Mannheim B2 , 1"  
route_df <- route(from, to, structure = "route")
```

Mehr Information

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

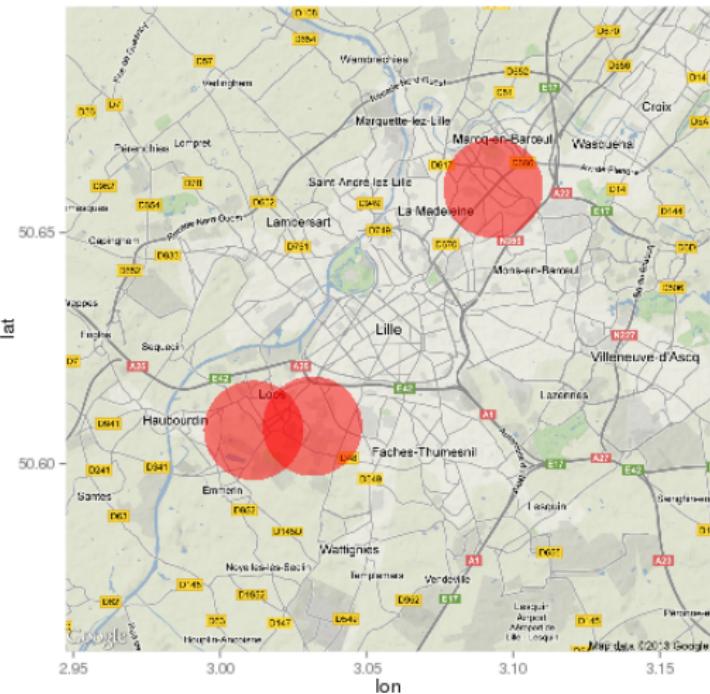
Eine Karte mit dieser Information zeichnen

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



Bubbles auf einer Karte

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



Cheatsheet

• Cheatsheet zu data visualisation

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

Data Visualization with ggplot2 Cheat Sheet

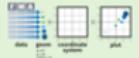


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data set**, a **set of `geom`s**—visual marks that represent data points, and a **coordinate system**.

ggplot

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



Build a graph with `ggplot()` or `ggplot2()`

`ggplot(mapping = ...)`
`ggplot(mapping = ...)`
`ggplot(mapping = ...)`
Creates a complete plot with given data, geom, and mapping. Supports many visual details.

`ggplot(data = mpg, aes(x = cyl, y = hwy))`
Begin a plot that you finish by adding layers to. No defaults, but provides more control than `plot()`.

`ggplot(mpg, aes(hwy, cyl)) +
 geom_point() +
 geom_smooth(method = "lm") +
 geom_text(aes(label = scales::label_percent())) +
 theme_bw()`
Add a layer to a plot with a `geom`, `stat`, or `stat`-`function`. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

`last_plot()`
Returns the last plot
`ggplot("plot.png", width = 5, height = 5)`

Jan-Philipp Kolb

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))`

`a + geom_area(stat = "bin")`

`a + geom_density(fill = "grey90")`

`a + geom_dotplot(binwidth = 0.5, fill = "white")`

`a + geom_freqpoly(binwidth = 1)`

`a + geom_histogram(binwidth = 1)`

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

`b + geom_bar()`

`b + geom_hex()`

`b + geom_linered()`

`b + geom_pointrange()`

`b + geom_rect()`

`b + geom_rug()`

`b + geom_smooth()`

`b + geom_text(label = "y")`

`b + geom_step(direction = "in")`

Discrete

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

`b + geom_bar()`

`b + geom_hex()`

`b + geom_linered()`

`b + geom_pointrange()`

`b + geom_rect()`

`b + geom_rug()`

`b + geom_smooth()`

`b + geom_text(label = "y")`

`b + geom_step(direction = "out")`

`b + geom_violin(scale = "count")`

`b + geom_voronoi()`

`b + geom_warm()`

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

`c + geom_bar(stat = "sum")`

`c + geom_hex(stat = "sum")`

`c + geom_linered(stat = "sum")`

`c + geom_pointrange(stat = "sum")`

`c + geom_rect(stat = "sum")`

`c + geom_rug(stat = "sum")`

`c + geom_smooth(stat = "sum")`

`c + geom_text(stat = "sum")`

`c + geom_step(stat = "sum")`

`c + geom_violin(stat = "sum")`

`c + geom_voronoi(stat = "sum")`

`c + geom_warm(stat = "sum")`

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

`d + geom_bar(stat = "sum")`

`d + geom_hex(stat = "sum")`

`d + geom_linered(stat = "sum")`

`d + geom_pointrange(stat = "sum")`

`d + geom_rect(stat = "sum")`

`d + geom_rug(stat = "sum")`

`d + geom_smooth(stat = "sum")`

`d + geom_text(stat = "sum")`

`d + geom_step(stat = "sum")`

`d + geom_violin(stat = "sum")`

`d + geom_voronoi(stat = "sum")`

`d + geom_warm(stat = "sum")`

`e <- ggplot(economics, aes(date, long))`

`e + geom_line()`

`e + geom_rect()`

`e + geom_rug()`

`e + geom_smooth()`

`e + geom_text()`

`e + geom_step()`

`e + geom_violin()`

`e + geom_voronoi()`

`e + geom_warm()`

`f <- ggplot(economics, aes(date, long))`

`f + geom_line()`

`f + geom_rect()`

`f + geom_rug()`

`f + geom_smooth()`

`f + geom_text()`

`f + geom_step()`

`f + geom_violin()`

`f + geom_voronoi()`

`f + geom_warm()`

`g <- ggplot(economics, aes(date, long))`

`g + geom_line()`

`g + geom_rect()`

`g + geom_rug()`

`g + geom_smooth()`

`g + geom_text()`

`g + geom_step()`

`g + geom_violin()`

`g + geom_voronoi()`

`g + geom_warm()`

`h <- ggplot(economics, aes(date, long))`

`h + geom_line()`

`h + geom_rect()`

`h + geom_rug()`

`h + geom_smooth()`

`h + geom_text()`

`h + geom_step()`

`h + geom_violin()`

`h + geom_voronoi()`

`h + geom_warm()`

`i <- ggplot(economics, aes(date, long))`

`i + geom_line()`

`i + geom_rect()`

`i + geom_rug()`

`i + geom_smooth()`

`i + geom_text()`

`i + geom_step()`

`i + geom_violin()`

`i + geom_voronoi()`

`i + geom_warm()`

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

`j + geom_line()`

`j + geom_rect()`

`j + geom_rug()`

`j + geom_smooth()`

`j + geom_text()`

`j + geom_step()`

`j + geom_violin()`

`j + geom_voronoi()`

`j + geom_warm()`

`k <- ggplot(economics, aes(date, long))`

`k + geom_line()`

`k + geom_rect()`

`k + geom_rug()`

`k + geom_smooth()`

`k + geom_text()`

`k + geom_step()`

`k + geom_violin()`

`k + geom_voronoi()`

`k + geom_warm()`

`l <- ggplot(economics, aes(date, long))`

`l + geom_line()`

`l + geom_rect()`

`l + geom_rug()`

`l + geom_smooth()`

`l + geom_text()`

`l + geom_step()`

`l + geom_violin()`

`l + geom_voronoi()`

`l + geom_warm()`

`m <- ggplot(economics, aes(date, long))`

`m + geom_line()`

`m + geom_rect()`

`m + geom_rug()`

`m + geom_smooth()`

`m + geom_text()`

`m + geom_step()`

`m + geom_violin()`

`m + geom_voronoi()`

`m + geom_warm()`

`n <- ggplot(economics, aes(date, long))`

`n + geom_line()`

`n + geom_rect()`

`n + geom_rug()`

`n + geom_smooth()`

`n + geom_text()`

`n + geom_step()`

`n + geom_violin()`

`n + geom_voronoi()`

`n + geom_warm()`

`o <- ggplot(economics, aes(date, long))`

`o + geom_line()`

`o + geom_rect()`

`o + geom_rug()`

`o + geom_smooth()`

`o + geom_text()`

`o + geom_step()`

`o + geom_violin()`

`o + geom_voronoi()`

`o + geom_warm()`

`p <- ggplot(economics, aes(date, long))`

`p + geom_line()`

`p + geom_rect()`

`p + geom_rug()`

`p + geom_smooth()`

`p + geom_text()`

`p + geom_step()`

`p + geom_violin()`

`p + geom_voronoi()`

`p + geom_warm()`

`q <- ggplot(economics, aes(date, long))`

`q + geom_line()`

`q + geom_rect()`

`q + geom_rug()`

`q + geom_smooth()`

`q + geom_text()`

`q + geom_step()`

`q + geom_violin()`

`q + geom_voronoi()`

`q + geom_warm()`

`r <- ggplot(economics, aes(date, long))`

`r + geom_line()`

`r + geom_rect()`

`r + geom_rug()`

`r + geom_smooth()`

`r + geom_text()`

`r + geom_step()`

`r + geom_violin()`

`r + geom_voronoi()`

`r + geom_warm()`

`s <- ggplot(economics, aes(date, long))`

`s + geom_line()`

`s + geom_rect()`

`s + geom_rug()`

`s + geom_smooth()`

`s + geom_text()`

`s + geom_step()`

`s + geom_violin()`

`s + geom_voronoi()`

`s + geom_warm()`

`t <- ggplot(economics, aes(date, long))`

`t + geom_line()`

`t + geom_rect()`

`t + geom_rug()`

`t + geom_smooth()`

`t + geom_text()`

`t + geom_step()`

`t + geom_violin()`

`t + geom_voronoi()`

`t + geom_warm()`

`u <- ggplot(economics, aes(date, long))`

`u + geom_line()`

`u + geom_rect()`

`u + geom_rug()`

`u + geom_smooth()`

`u + geom_text()`

`u + geom_step()`

`u + geom_violin()`

`u + geom_voronoi()`

`u + geom_warm()`

`v <- ggplot(economics, aes(date, long))`

`v + geom_line()`

`v + geom_rect()`

`v + geom_rug()`

`v + geom_smooth()`

`v + geom_text()`

`v + geom_step()`

`v + geom_violin()`

`v + geom_voronoi()`

`v + geom_warm()`

`w <- ggplot(economics, aes(date, long))`

`w + geom_line()`

`w + geom_rect()`

`w + geom_rug()`

`w + geom_smooth()`

`w + geom_text()`

`w + geom_step()`

`w + geom_violin()`

`w + geom_voronoi()`

`w + geom_warm()`

`x <- ggplot(economics, aes(date, long))`

`x + geom_line()`

`x + geom_rect()`

`x + geom_rug()`

`x + geom_smooth()`

`x + geom_text()`

`x + geom_step()`

`x + geom_violin()`

</div

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:

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

Die lineare Regression

Die lineare Regression

Maindonald - Data Analysis

- 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|)  
##
```

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 '
```

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
## 2.979669 6.179765 6.713114 10.713233 12.046606 14.180002
##          8          9         10
## 18.180121 24.046962 30.980502
```

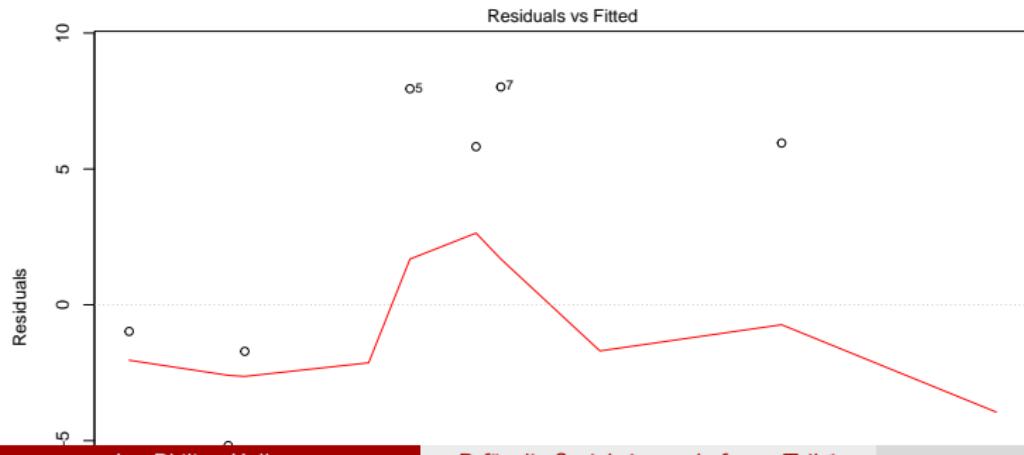
```
resid(roller.lm) # Residuen
```

```
##          1          2          3          4          5          6
## -0.9796695 -5.1797646 -1.7131138 -5.7132327  7.9533944  5.8
##          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

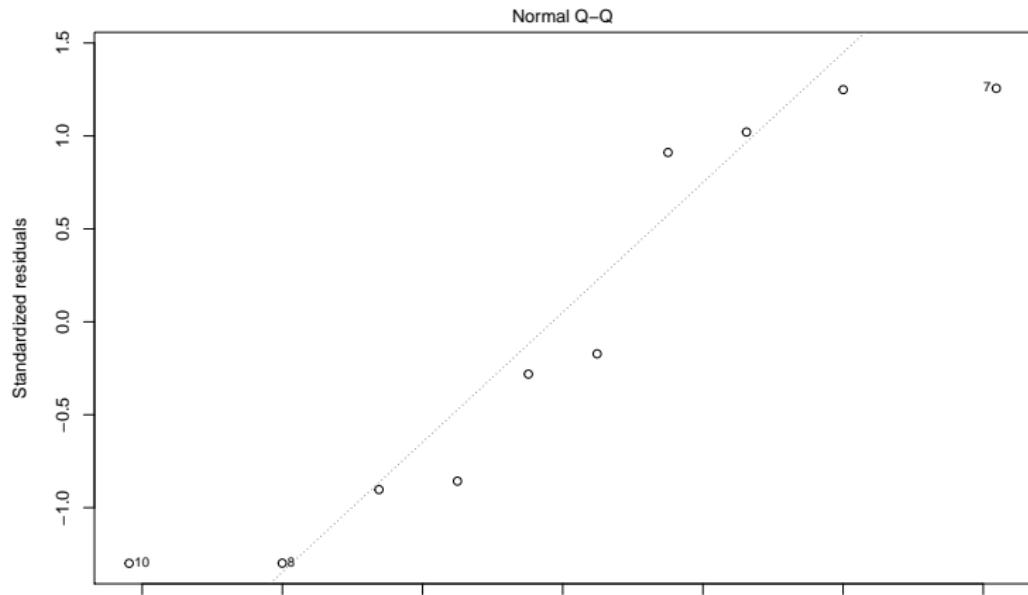
```
plot(roller.lm, 1)
```



Residuenplot

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

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



Regressionsdiagnostik mit Basis-R

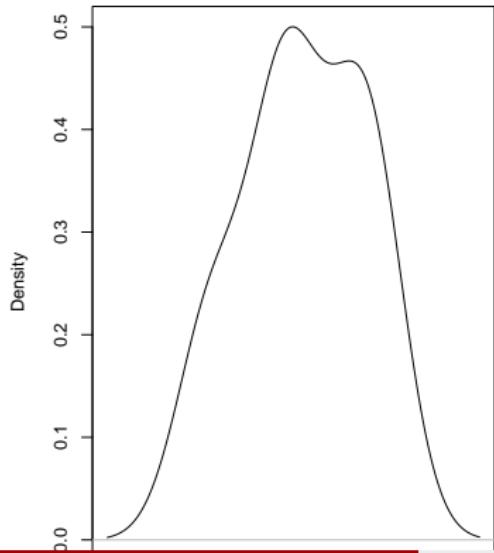
Ein einfaches Modell

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

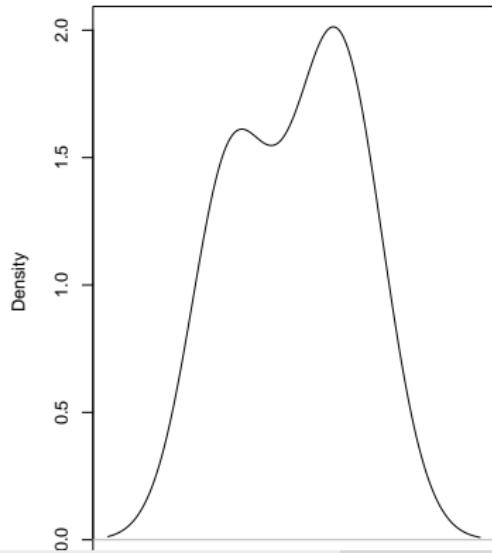
Die Dichte der beiden Vektoren

```
par(mfrow=c(1,2))  
plot(density(x1))  
plot(density(y))
```

`density.default(x = x1)`



`density.default(x = y)`



Modellvorhersage machen

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

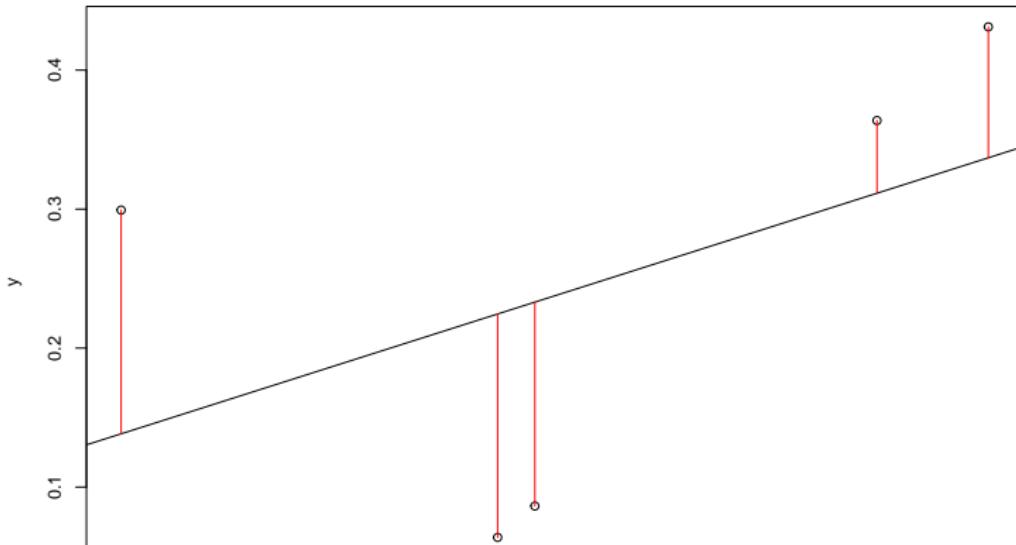
## [1] 0.08638181 0.29930752 0.43115604 0.06377895 0.36373294

pre

##          1           2           3           4           5
## 0.2330881 0.1383326 0.3369425 0.2245584 0.3114357
```

Regressionsdiagnostik mit Basis-R

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



Beispieldaten Luftqualität

```
library(datasets)
?airquality
```

```
airquality {datasets}
```

New York Air Quality Measurements

Description

Daily air quality measurements in New York, May to September 1973.

Usage

```
airquality
```

Format

A data frame with 154 observations on 6 variables.

```
[,1] Ozone    numeric Ozone (ppb)
[,2] Solar.R  numeric Solar R (lang)
[,3] Wind     numeric Wind (mph)
[,4] Temp     numeric Temperature (degrees F)
[,5:12] Month   numeric Month (1-12)
```

Das visreg-Paket

Ein Modell wird auf dem airquality Datensatz geschätzt

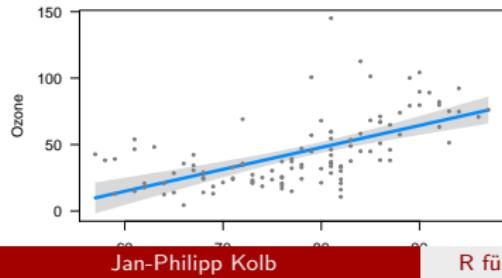
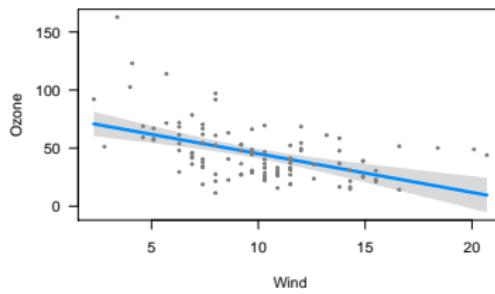
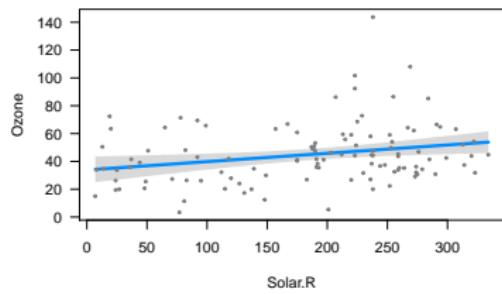
```
install.packages("visreg")

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

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind + Temp, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -40.485 -14.219 - 3.551  10.097  95.619 
##
## Coefficients:
```

Visualisierung

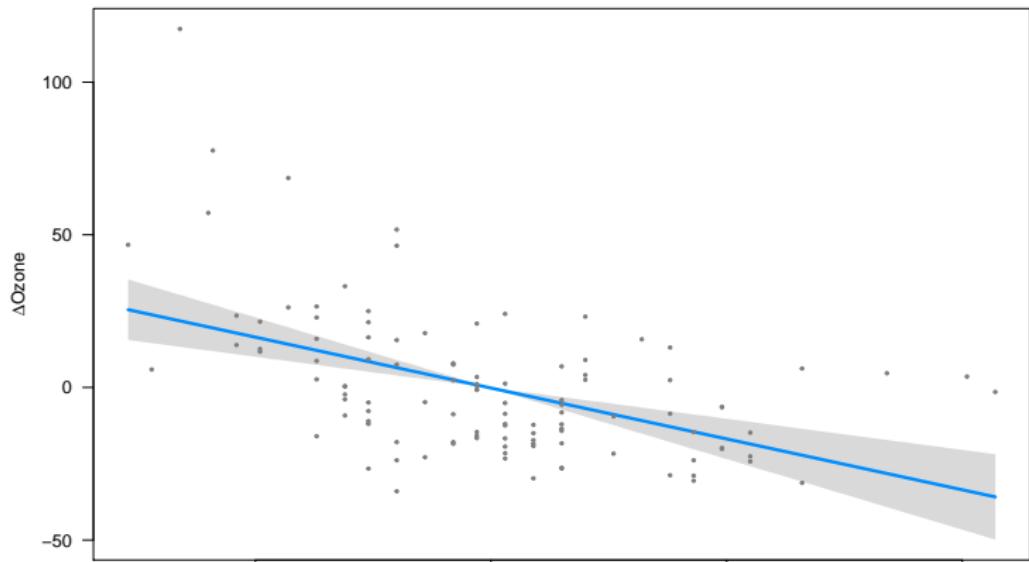
```
par(mfrow=c(2,2))  
visreg(fit)
```



Und dann mit visreg visualisiert.

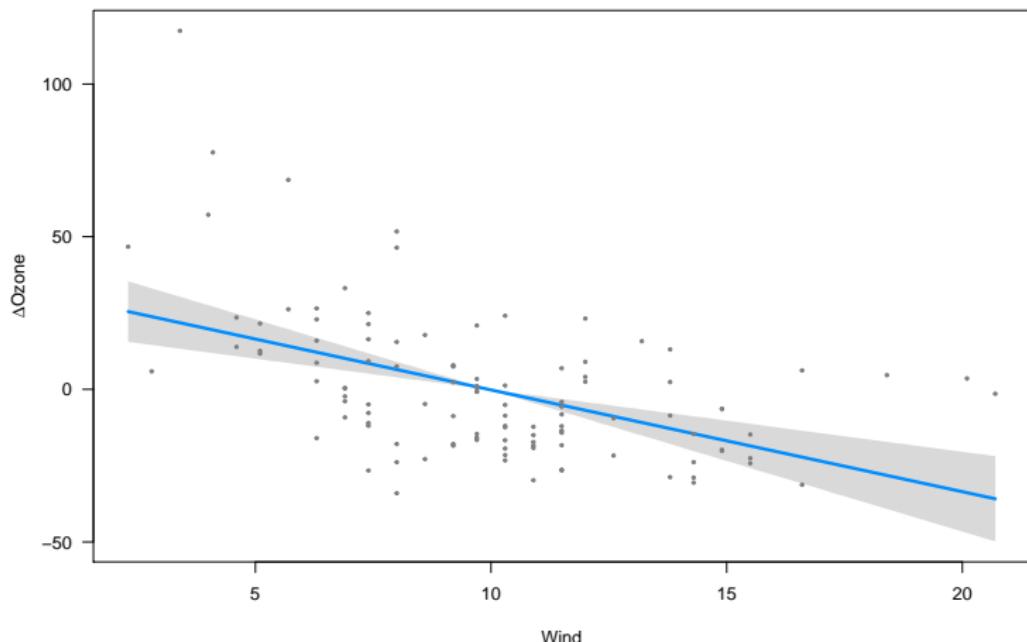
- Zweites Argument - Spezifikation erklärende Variable für Visualisierung

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



Visualisierung mit dem Paket visreg

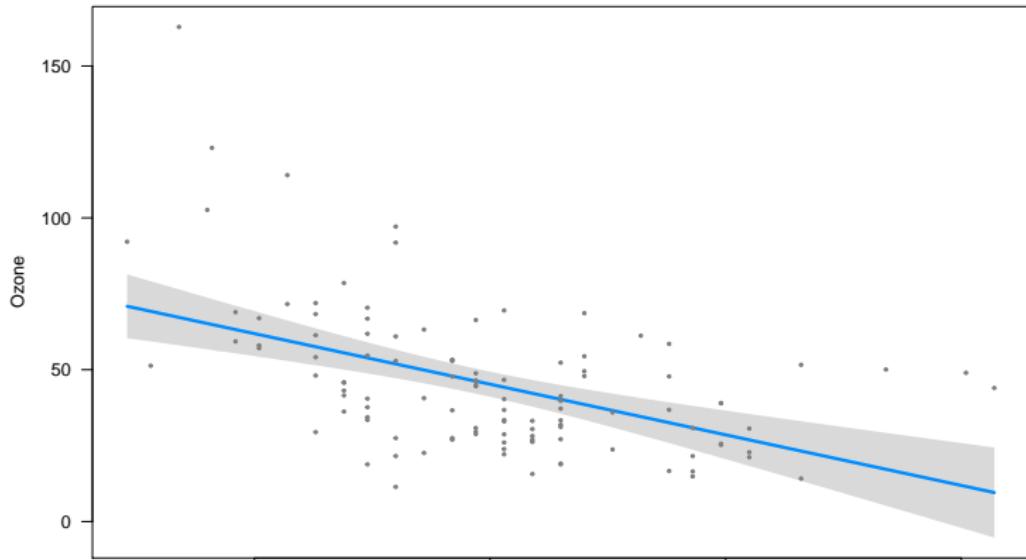
```
visreg(fit, "Wind", type = "contrast")
```



Das visreg-Paket

- Das Default-Argument für type ist conditional.

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



Regression mit Faktoren

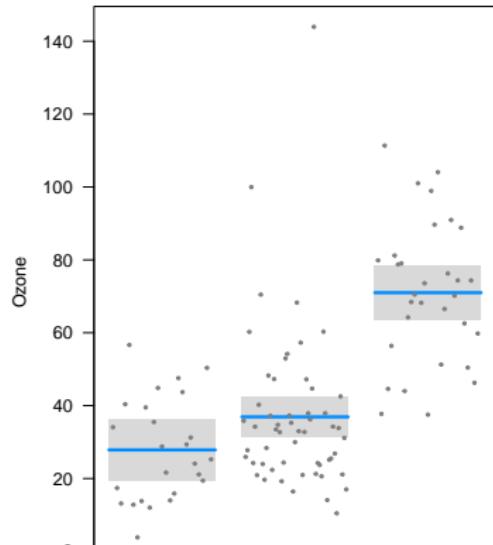
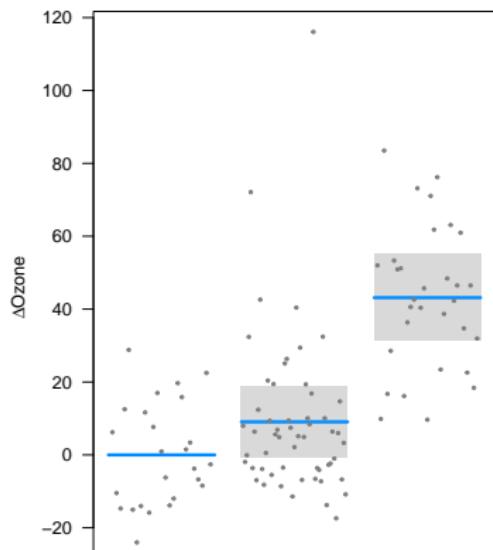
Mit visreg können die Effekte bei Faktoren visualisiert werden.

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

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind + Heat, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -33.473 -12.794  -2.686   8.461 107.035
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.8851    1.8778  20.158  <2e-16 ***
## Solar.R     0.1239    0.0172   7.208  1.51e-09 ***
## Wind        0.0083    0.0018   4.618  1.02e-05 ***
## Heat        0.0000    0.0000    0.000    1.0000
##
```

Effekte von Faktoren

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



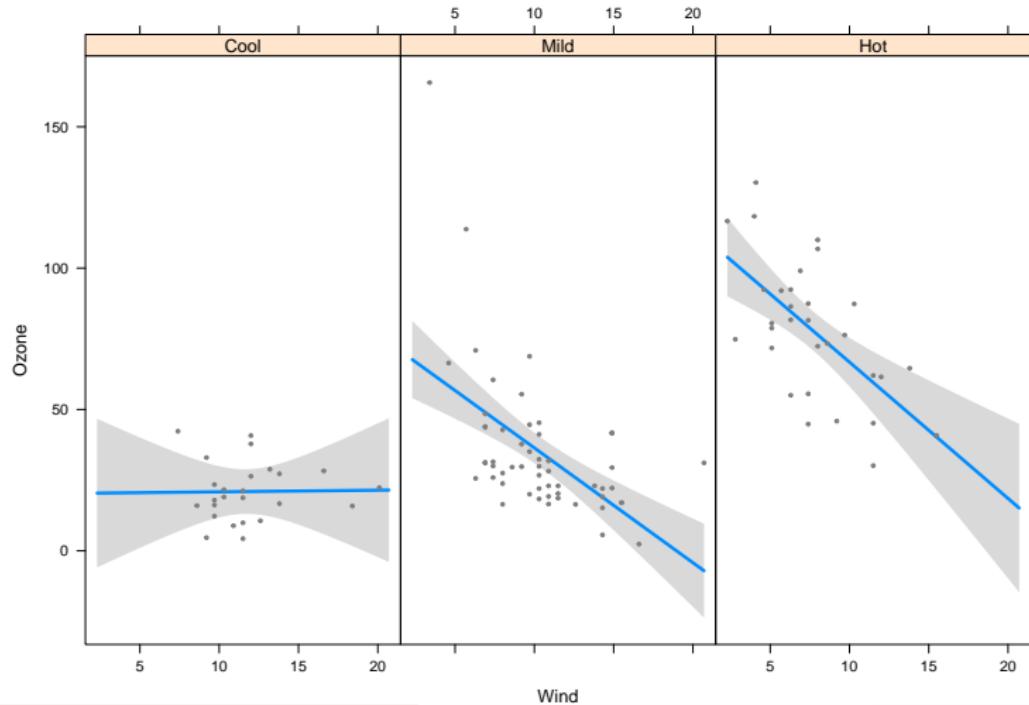
Das Paket visreg - Interaktionen

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

##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind * Heat, data = airquality)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -34.472 -11.640  -1.919   7.403 102.428
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.48042   17.38219   0.258 0.797102
```

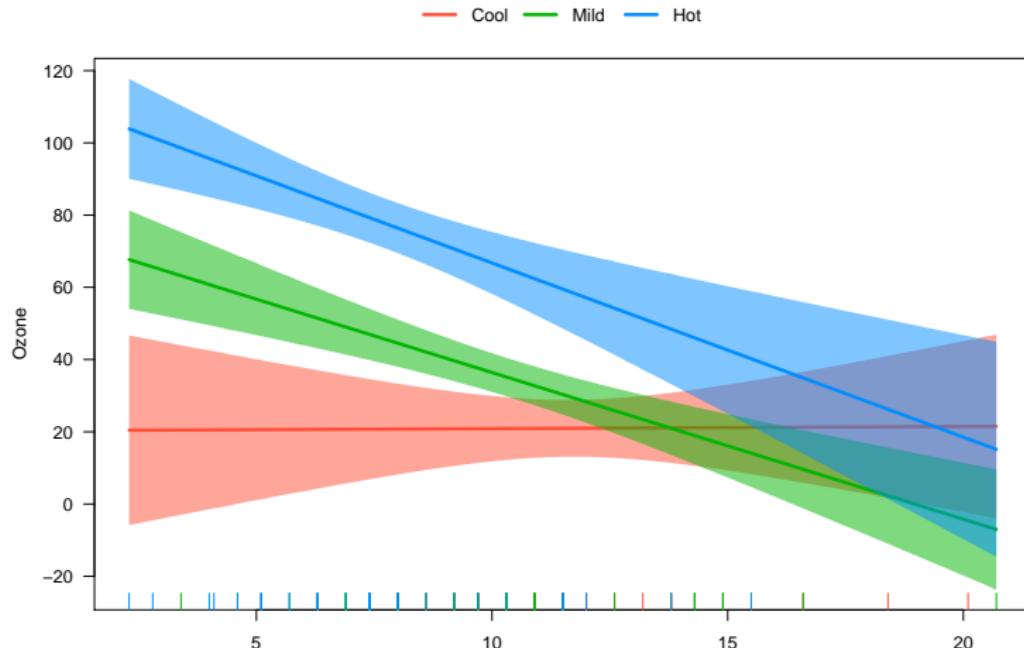
Steuern der Graphikausgabe mittels layout

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



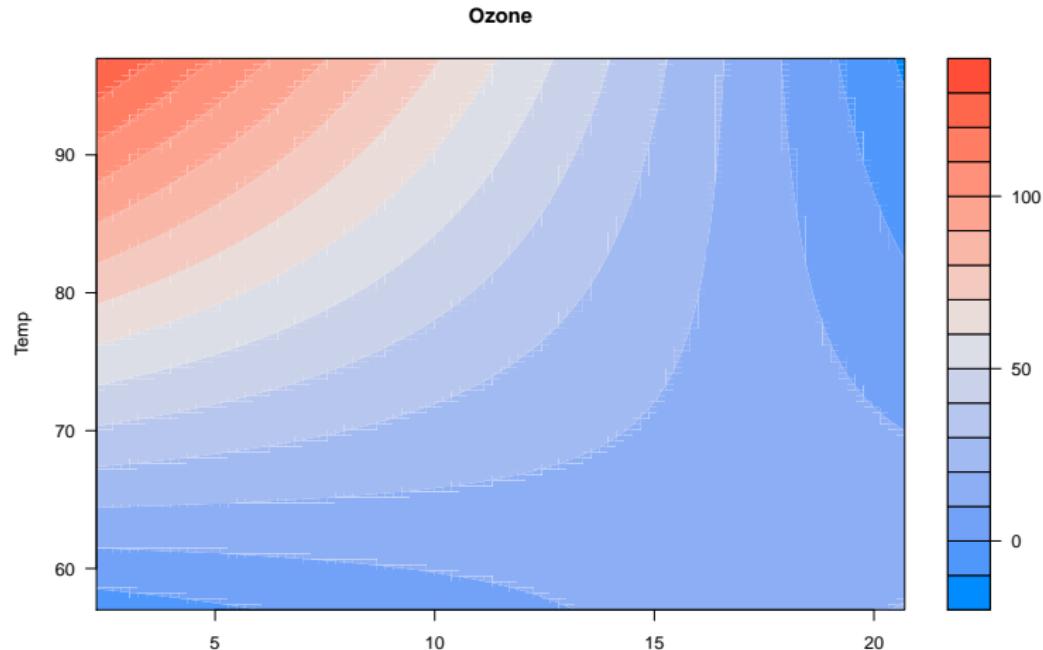
Das Paket visreg - Interaktionen overlay

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



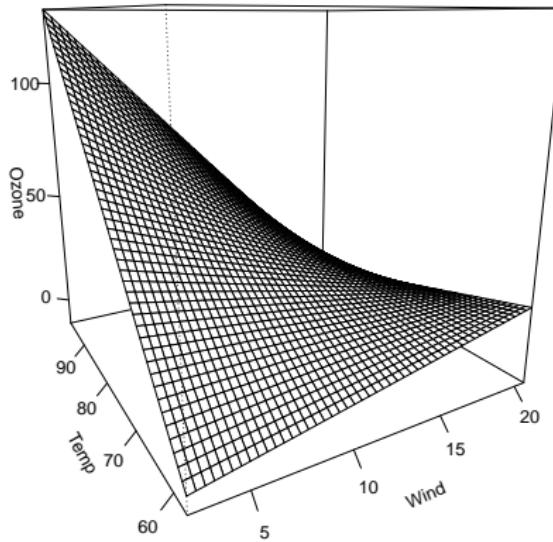
Das Paket visreg - visreg2d

```
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")
```



Regression mit dem survey Paket

```
library(survey)
data(api)
head(api.pop)

##          cds  stype         name
## 1 01611190130229      H   Alameda High
## 2 01611190132878      H   Encinal High
## 3 01611196000004      M Chipman Middle
## 4 01611196090005      E Lum (Donald D.) Lum (Donald D.) Elem
## 5 01611196090013      E Edison Elementa
## 6 01611196090021      E Otis (Frank) El    Otis (Frank) Elem
##          dname  dnum   cname  cnum flag pcttest api00
## 1 Alameda City Unified 6 Alameda 1 NA 96 731
## 2 Alameda City Unified 6 Alameda 1 NA 99 622
## 3 Alameda City Unified 6 Alameda 1 NA 99 622
## 4 Alameda City Unified 6 Alameda 1 NA 99 774
```

Das Survey Design spezifizieren

```
dstrat<-svydesign(id=~1,strata=~stype, weights=~pw,
                    data=apistrat, fpc=~fpc)
```

Die Regression rechnen

```
summary(svyglm(api00~ell+meals+mobility,
                design=dstrat))
```

```
##  
## Call:  
## svyglm(formula = api00 ~ ell + meals + mobility, design = dstrat)  
##  
## Survey design:  
## svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat,  
##           fpc = ~fpc)  
##  
## Coefficients:
```

Linkliste - lineare Regression

- Regression - r-bloggers
- Das Komplette Buch von Faraway- sehr intuitiv geschrieben.
- Gute Einführung auf Quick-R
- Multiple Regression
- Basis Regression - How to go about interpreting regression coefficients

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)
- ① Lesen Sie den Datensatz `toycars` (Paket DAAG) in einen dataframe `data` ein und wandeln Sie die Variable `car` des Datensatzes in einen Faktor (`as.factor`) um.
 - ② Erstellen Sie drei Boxplots, die die zurückgelegte Strecke getrennt nach dem Faktor `car` darstellen.
 - ③ Schätzen Sie für die Autos die Parameter des folgenden linearen Modells mit Hilfe der Funktion `lm()`

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