

R für die Sozialwissenschaften - Teil 1

Jan-Philipp Kolb

22 Juni 2017

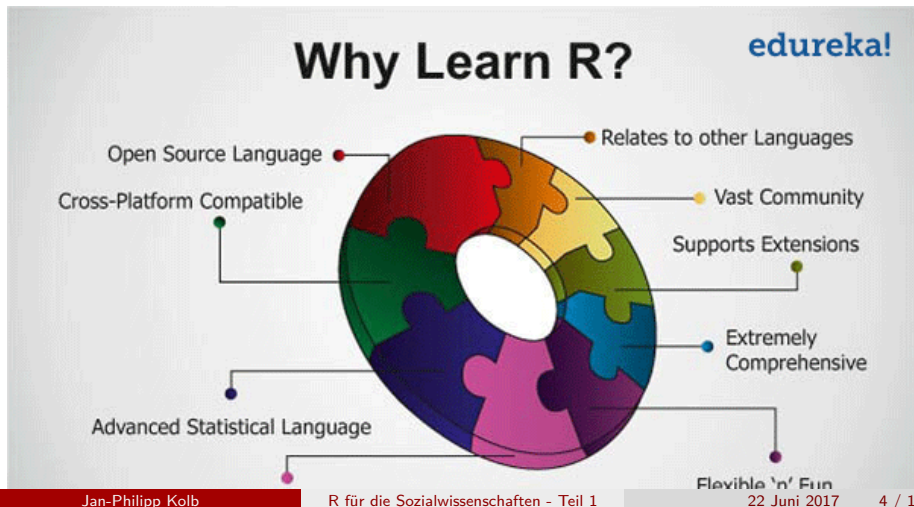
Einführung und Motivation

Pluspunkte von R

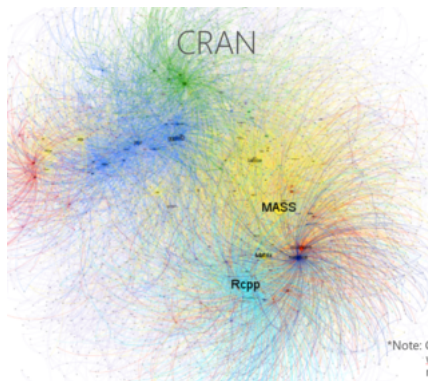
- Als Weg kreativ zu sein ...
- Graphiken, Graphiken, Graphiken
- In Kombination mit anderen Programmen nutzbar
- Zur Verbindung von Datenstrukturen
- Zum Automatisieren
- Um die Intelligenz anderer Leute zu nutzen ;-)
- ...

Gründe

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

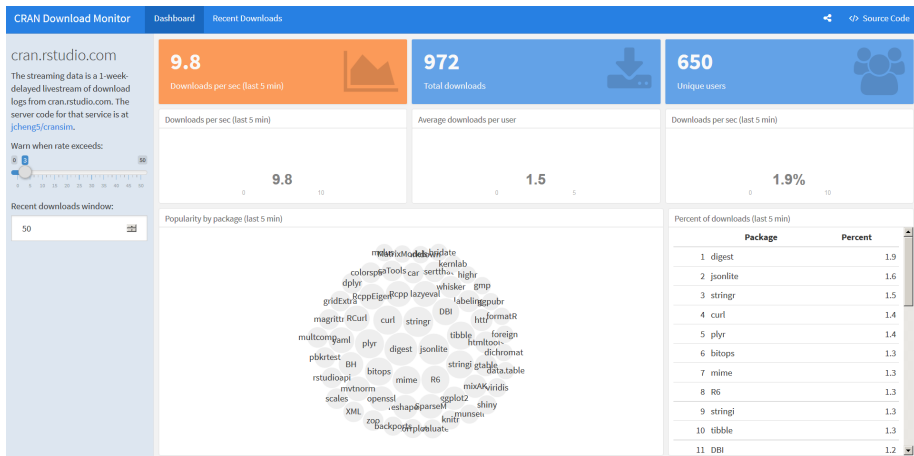


Modularer Aufbau



*Note: Colour indicates communities found by the walktrap algorithm, but has no common meaning in the two networks

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

- <https://japhilko.github.io/RSocialScience/>



The screenshot shows the homepage of the RSocialScience website. The header is blue with the title 'RSocialScience' and the subtitle 'Kurs zur Nutzung von R in den Sozialwissenschaften'. Below the header is a yellow navigation bar with a 'Home' button and a GitHub logo. The main content area is light beige and features a sidebar on the left with a list of topics: 'Einführung', 'Liebe auf den ersten Plot – Grafiken und Datenanalyse mit R', 'Regression mit R', 'Arbeitsorganisation mit Rstudio und git', and 'Präsentation von Daten – Reproducible Research'. The main content area has a section titled 'Einführung' with a list of topics: 'Einführung und Motivation', 'Erste Schritte mit R', 'Wie bekommt man Hilfe?', 'Modularer Aufbau', 'Datenimport', 'Datenaufbereitung', and 'Datenexport'.

RSocialScience
Kurs zur Nutzung von R in den Sozialwissenschaften

Home

Einführung

Liebe auf den ersten Plot –
Grafiken und Datenanalyse mit
R

Regression mit R

Arbeitsorganisation mit
Rstudio und git

Präsentation von Daten –
Reproducible Research

Einführung

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

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 - Die Arbeitsorganisation mit Rstudio und Rmarkdown
- Teil 4 - 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/RSocialScience/tree/master/code>

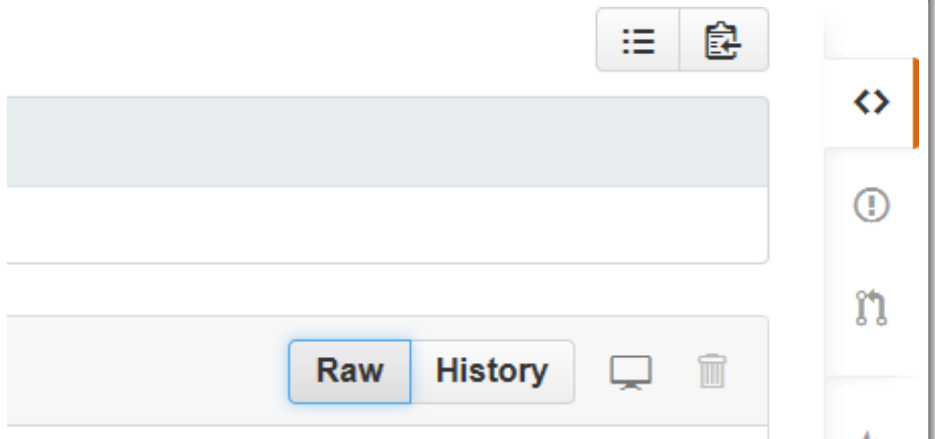
Daten herunterladen

- 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 auch 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.

The screenshot shows a Mac OS X desktop with two windows. The background window is the 'R Console' with the title bar 'Grab File Edit Capture Window Help'. The menu bar shows 'ma 08:24'. The console text includes the R version (2.8.0), copyright (2008), and a list of masked objects from the 'coda' package. It also shows the loading of the 'arm' package and the execution of a series of commands to read data, create a model, and save output. The foreground window is a 'Quartz 2 [R]' plot window showing a scatter plot of 'c(1,2,3)' on the y-axis (ranging from 1.0 to 3.0) against 'Index' on the x-axis (ranging from 1.0 to 3.0). The plot contains three points at (1,1), (2,2), and (3,3).

```
R version 2.8.0 (2008-10-20)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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))
>
```

```
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'

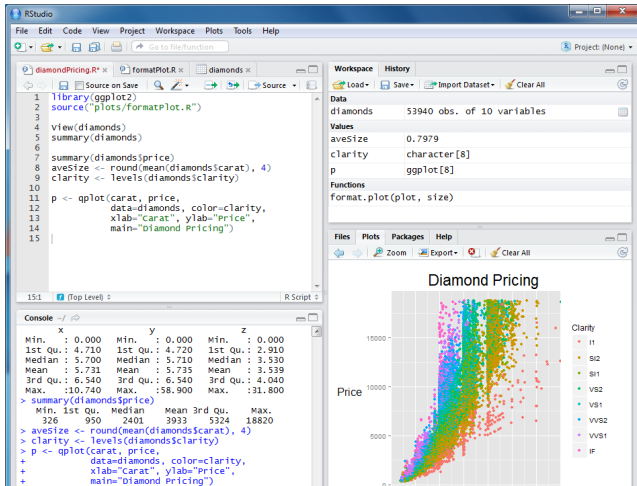
The following object(s) are masked from package:coda :

  traceplot

> dat.stijn <- read.table("/Users/Rense/Documents/RWork/stijndata.txt")
>
> names(dat.stijn) <- c("cons", "cntrywave", "aantal", "respnr", "wave",
+ "country", "sex", "agec", "agesq", "educ",
+ "married", "cohabit", "separate", "widow", "single",
+ "cath", "prot", "other", "none", "attendance",
+ "volund", "mlavattend", "gdpnormal", "gastil_r", "cwave",
+ "numcountry", "na1", "na2", "na3", "na4")
>
> model.stijn <- glm(c(1,2,3) ~ sex + educ + agec + agesq + cohabit + separate + widow + single + cath +
+ prot + other + attendance + mlavattend + 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



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] "Tue Jun 27 17:26:49 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
length()	Länge	length(b)
max()	Maximum	max(b)
min()	Minimum	min(b)
sd()	Standardabweichung	sd(b)
var()	Varianz	var(b)
mean()	Mittelwert	mean(b)
median()	Median	median(b)

Diese Funktionen brauchen nur ein Argument.

Funktionen mit mehr Argumenten

Andere Funktionen brauchen mehr:

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

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 reele Zahlen	5, 3.462
logical	logische Werte	FALSE, TRUE
character	Buchstaben und Zeichenfolgen	"Hallo"

Quelle: R. Munnich und M. Knobelspieß (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 1 10 mal
```

```
rep(1,10)
```

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

```
rep("A",10)
```

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

Die Funktion paste

```
?paste
```

```
paste(1:4)
```

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

```
paste("A", 1:6, sep = "")
```

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

- 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 `?` um Hilfe zu bekommen.

```
?mean
```

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

```
example(lm)
```

- Dokumente zur Veranschaulichung und Erläuterung von Funktionen im Paket

```
browseVignettes()
```

- 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.b
## [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

 **stackoverflow**

Questions Jobs Documentation BETA Tags Users

  [Log In](#) [Sign Up](#)

Tagged Questions

info newest **8** featured frequent votes active unanswered

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. Use `dput()` for data and specify all non-base packages with library calls. For statistical questions ...

[learn more...](#) [top users](#) [synonyms \(2\)](#) [r jobs](#)

1776 votes

22 answers

147k views

[r](#) [r-faq](#)


How to make a great R reproducible example?

When discussing performance with colleagues, teaching, sending a bug report or searching for guidance on mailing lists and here on SO, a reproducible example is often asked and always helpful. What ...

community wiki
11 revs, 8 users 54%
[Hack-R](#)

22,187 frequent questions tagged

[r](#) [about »](#)

 **R Language**
DOCUMENTATION

[Find a request to handle](#) or [browse 121 topics](#).

Related Tags

[ggplot2](#) × 2875

[dataframe](#) × 1351

[plot](#) × 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

?mean
Get help of a particular function.

help.search("weighted.mean")
Search the help files for a word or phrase.

help(package = "dplyr")
Find help for a package.

More about an object

str(object)
Get a summary of an object's structure.

class(object)
Find the class an object belongs to.

Using Libraries

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

library(dplyr)
Load the package into the session, making all R/Functions available to use.

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

detach()
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/you/path")
Change the current working directory.

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

Vectors

Creating Vectors

<code>c(1, 4, 8)</code>	<code>1 4 8</code>	Arithmetic vector
<code>1:4</code>	<code>1 2 3 4</code>	Integer sequence
<code>rep(1, 4, length=4)</code>	<code>1 1 1 1</code>	A complex sequence
<code>rep(1:4, times=2)</code>	<code>1 2 3 4 1 2 3 4</code>	Repeating vector
<code>rep(1:4, each=2)</code>	<code>1 1 2 2 3 3 4 4</code>	Repeating elements

Vector Functions

<code>sort(x)</code>	Return sorted
<code>unique(x)</code>	Return unique values
<code>length(x)</code>	Get length of vector

Selecting Vector Elements

By Position

<code>x[4]</code>	The fourth element
<code>x[-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 %>% 2]</code>	Elements in the set 1, 2, 5

Named Vectors

<code>x["apple"]</code>	Element with name 'apple'
-------------------------	---------------------------

Programming

For Loop

```
for (something in something) {  
  do something  
}
```

Example

```
for (i in 1:10) {  
  x[i] = 5 * i  
  print(i)  
}
```

While Loop

```
while (condition) {  
  do something  
}
```

Example

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

If Statements

```
if (condition) {  
  do something  
} else {  
  do something different  
}
```

Example

```
if (i < 5) {  
  print("Yes")  
} else {  
  print("No")  
}
```

Functions

```
function_name <- function(parameters) {  
  do something  
  return(something_variable)  
}
```

Example

```
squares <- function(x) {  
  squared <- x^2  
  return(squared)  
}
```

Reading and Writing Data

Input	Output	Description
<code>df <- read.table("file.csv")</code>	<code>write.table(df, "file.csv")</code>	Read and write a delimited text file.
<code>df <- read.csv("file.csv")</code>	<code>write.csv(df, "file.csv")</code>	Read and write a comma-separated value file. This is a special case of read.table() assuming
<code>load("file.Rsave")</code>	<code>save(df, file = "file.Rsave")</code>	Read and write an R data file, a R-specific format.

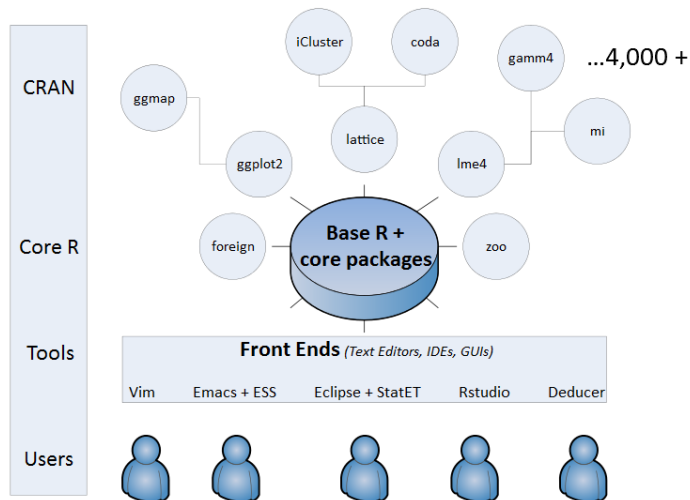
condition	0 <= 0	Not equal	0 > 0	Greater than	0 <= 0	Greater than or equal to	0 < 0	Less than	0 >= 0	Less than or equal to
	0 < 0	Not equal	0 < 0	Greater than	0 < 0	Greater than or equal to	0 < 0	Less than	0 >= 0	Less than or equal to

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

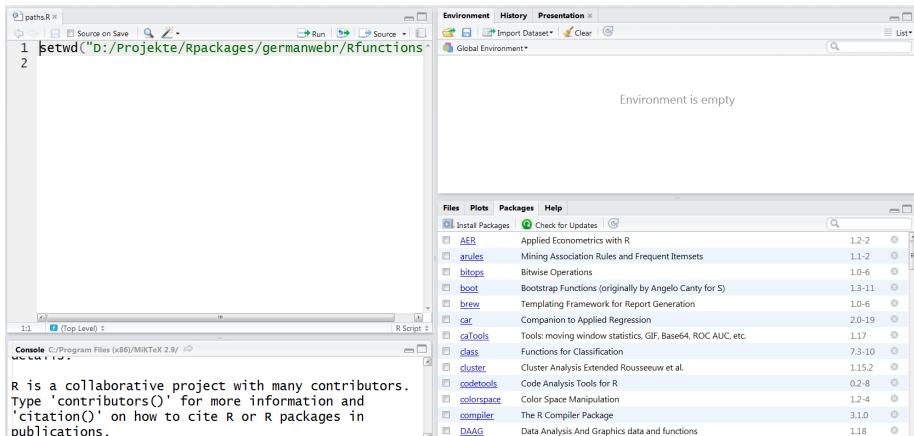


Installation

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

```
library(lme4)
```


Installation von Paketen mit RStudio



The screenshot displays the RStudio interface with the following components:

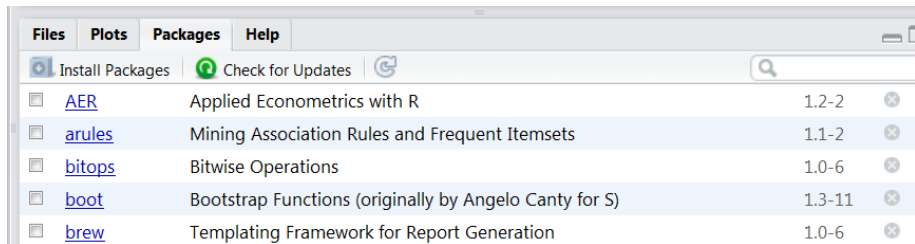
- Script Editor:** Contains the R code:

```
1 setwd("D:/Projekte/Rpackages/germanwehr/Rfunctions")  
2
```
- Environment Pane:** Shows 'Global Environment *' and the message 'Environment is empty'.
- Packages Pane:** Displays a list of installed and available packages. The 'Install Packages' tab is active, showing a table of packages.

Package	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

Console: Displays the message: '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.'

Vorhandene Pakete und Installation



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

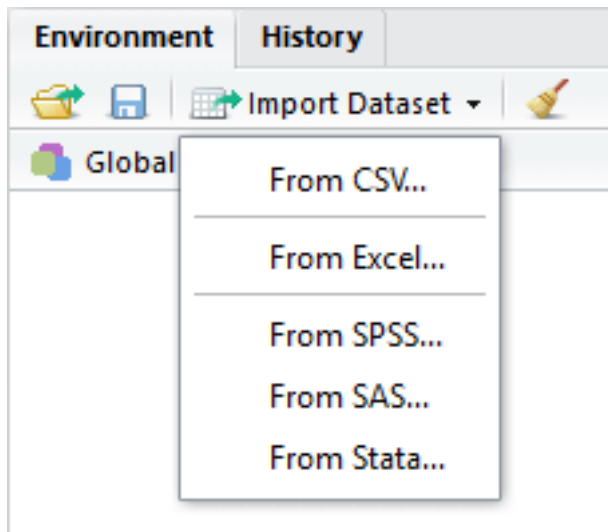
- Von R werden quelloffene, nicht-proprietäre Formate bevorzugt
- Es können aber auch Formate von anderen Statistik Software Paketen eingelesen werden
- R-user speichern Objekte gerne in sog. Workspaces ab
- Auch hier jedoch gilt: (fast) alles andere ist möglich

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 Text Data

File/URL

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

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	\$68876.16	\$72336.79	NA	POL	Department of Police	MIS Information Management and Technology Divisi...	Fulltime-Regular	Office Services Coordinator	NA
Aaron, David J.	M	\$96908.09	\$101857.00	\$6640.99	POL	Department of Police	ISB Major Crimes Division Fugitive Section	Fulltime-Regular	Master Police Officer	NA
Aaron, Marsha M.	F	\$104196.06	\$103019.73	NA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Fulltime-Regular	Social Worker IV	NA
Abalos, Coffred A.	M	\$50697.79	\$54181.46	\$6445.15	COR	Correction and Rehabilitation	PRRS Facility and Security	Fulltime-Regular	Resident Supervisor II	NA
Ababu, Essayas	M	\$59391.00	\$59468.35	NA	HCA	Department of Housing and Community Affairs	Single Family Housing Program	Fulltime-Regular	Planning Specialist III	NA
Abdmonem, Drea B.	M	\$67715.00	\$81392.40	\$10027.11	POL	Department of Police	PSB 6th District Special Assignment Team	Fulltime-Regular	Police Officer III	NA
Abdelmonem, Marwan M.	M	\$62286.30	\$59663.27	NA	HHS	Department of Health and Human Services	Head Start	Fulltime-Regular	Administrative Specialist II	NA
Abdul-Chan, Hasiinah J.	F	\$45828.92	\$46783.23	\$6.38	POL	Department of Police	PSB Traffic Division Automated Traffic Enforcement S...	Fulltime-Regular	Police Aide	NA
Abduljabbar, Saeed	M	\$61040.57	\$66861.98	\$6569.81	DCS	Department of General Services	Facilities Maintenance	Fulltime-Regular	Electrician I	NA
Abdur-Rahem, Mikael A.	M	\$56404.96	\$71943.08	\$15342.84	DOT	Department of Transportation	Transit Silver Spring Ride On	Fulltime-Regular	Bus Operator	NA
Abene, Hiruth	F	\$151585.60	\$164945.06	NA	HHS	Department of Health and Human Services	STD and HIV Services	Parttime-Regular	Medical Doctor III - Physician	NA
Abene, Zekarias S.	M	\$44825.99	\$51693.47	\$5240.75	DOT	Department of Transportation	Transit Nicholson Ride On	Fulltime-Regular	Bus Operator	NA
Abedin, Amiraziz	M	\$39062.00	\$450.00	NA	DOT	Department of Transportation	Transportation Management	Fulltime-Regular	Traffic Management Technician II	Traffic Management Technici...
Abelove, Sherry R.	F	\$93436.50	\$90833.10	NA	HHS	Department of Health and Human Services	Adult Protective and Case Management Services	Fulltime-Regular	Social Worker III	NA
Aben, Yoseph M.	M	\$117811.00	\$115786.22	NA	DTS	Department of Technology Services	EASD - ERP Applications Support	Fulltime-Regular	Senior Information Technology Specialist	NA
Abi Jamaa, Rania F.	F	\$53009.99	\$46850.36	NA	LIB	Department of Public Libraries	Olney Library	Fulltime-Regular	Library Assistant I	NA
Abijomas, Ryan Z.	M	\$17288.00	\$14655.53	NA	LIB	Department of Public Libraries	Silver Spring Library	Parttime-Regular	Library Desk Assistant	NA
Abita, Lydia B.	F	\$40429.58	\$41746.34	\$5500.53	DOT	Department of Transportation	Transit Caldersburg Ride On	Fulltime-Regular	Bus Operator	NA
Abikarian, Maral	F	\$20925.51	\$9976.52	\$45.28	POL	Department of Police	PSB Traffic Division School Safety Section	Parttime-Regular	Crossing Guard	NA
Abouraya, Nadia L.	F	\$16602.01	\$15592.92	NA	HHS	Department of Health and Human Services	Community Support Network for People with Disabilities	Parttime-Regular	Office Clerk	NA

Previewing first 50 entries.

Der Arbeitsspeicher

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

```
getwd()
```

So kann man das Arbeitsverzeichnis ändern:

Man erzeugt ein Objekt in dem man den Pfad abspeichert:

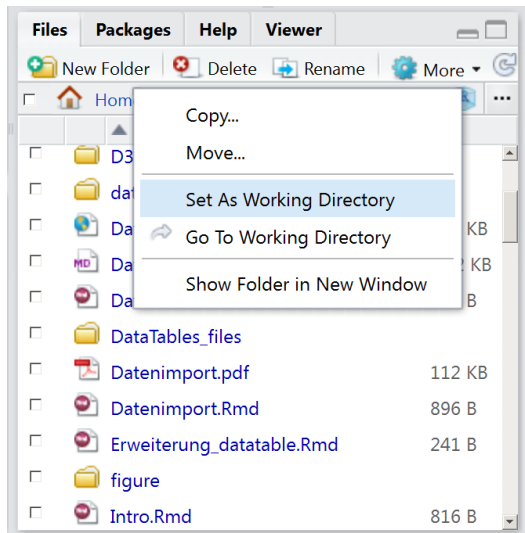
```
main.path <- "C:/" # Beispiel für Windows  
main.path <- "/users/Name/" # Beispiel für Mac  
main.path <- "/home/user/" # Beispiel für Linux
```

Und ändert dann den Pfad mit `setwd()`

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

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

Alternative - Arbeitsspeicher



Import von Excel-Daten

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

```
library(foreign)
```

```
?read.csv
```

```
?read.csv2
```


CSV Dateien einlesen

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

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

Wenn es sich um Deutsche Daten handelt:

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

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))
```

```
##                               whcSit  
## 1 Cultural Landscape and Archaeological Remains of the Bami  
## 2                               Minaret and Archaeological Rema  
## 3                               Historic Centres of Berat and G  
## 4  
## 5                               Al Qal'a of F  
## 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 angegeben 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 rio

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

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

Datenmanagement ähnlich wie in SPSS oder Stata

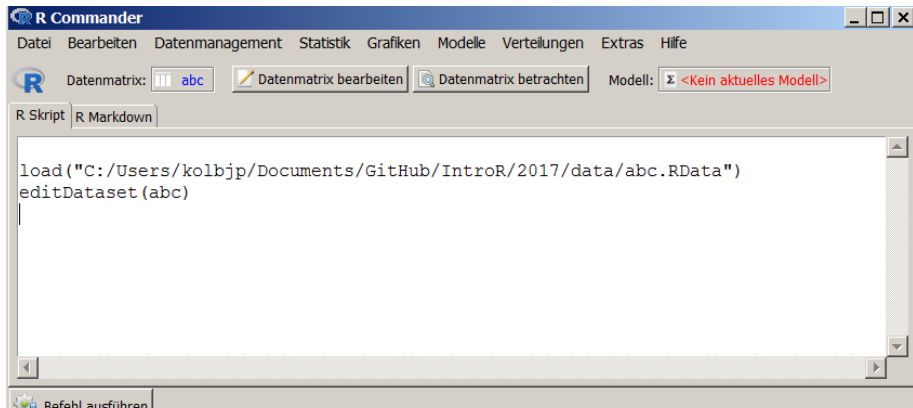
```
install.packages("Rz")  
library(Rz)
```

Weitere Alternative Rcmdr

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

- Funktioniert auch mit Rstudio

```
library(Rcmdr)
```



Aufgabe

- Gehen Sie auf meine Github Seite

`https://github.com/Japhilko/RSocialScience/tree/master/data`

- Importieren Sie den Datensatz `GPanel.dta`

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
```

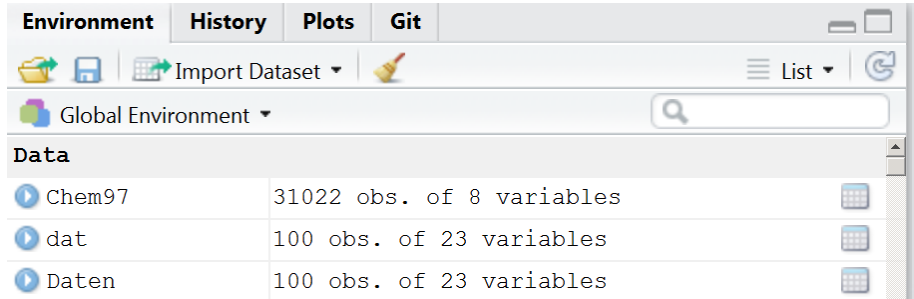
```
## [1] 23
```

Die Daten anschauen

- die ersten zeilen anschauen

`head(Daten)`

- Übersicht mittels Rstudio



The screenshot shows the RStudio interface's Environment pane. At the top, there are tabs for 'Environment', 'History', 'Plots', and 'Git'. Below these tabs is a toolbar with icons for file operations and a search bar. The main area of the pane is titled 'Global Environment' and contains a list of objects in the environment. The objects are 'Chem97', 'dat', and 'Daten'. Each object is represented by a blue play button icon, its name, and a description of its size and variables. To the right of each description is a small icon representing the object's class (e.g., data frame, matrix).

Data	
▶ Chem97	31022 obs. of 8 variables
▶ dat	100 obs. of 23 variables
▶ Daten	100 obs. of 23 variables

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
```

```
##          a11c034a b3g0020a a11c054a
```

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 un
## 2      Eher zufrieden          Sehr
## 3      Eher zufrieden          Eher
## 4      Sehr zufrieden          Eher
## 5      Sehr zufrieden          Sehr
## 6      Eher zufrieden          Sehr
## 7      Sehr zufrieden          Sehr
## 8      Sehr zufrieden          Sehr
## 9      Eher zufrieden          Eher
## 10     Sehr zufrieden          Eher
## 11     Eher zufrieden          Eher
```

Datensätze einschränken

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

```
Daten[SEX=="Männlich",]
```

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

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 un
```

```
## 3                            Eher zufrieden                            Sehr
```

```
## 5                            Eher zufrieden                            Eher
```

```
## 9                            Sehr zufrieden                            Eher
```

```
## 15                           Sehr zufrieden                            Sehr
```

```
## 18                           Eher zufrieden                            Sehr
```

```
## 20                           Eher zufrieden                            Eher
```

```
## 29                           Sehr zufrieden                            Sehr
```

```
## 41                           Sehr zufrieden                            Eher
```

Die Nutzung einer Sequenz

Daten[1:3,]

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

Variablen Labels

```
library(foreign)
dat <- read.dta("https://github.com/Japhilko/RSocialScience/blob/master/data/1999.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/1999.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 zufried  
## [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 zufried  
## [5] Eher zufrieden Sehr zufrieden Eher zufrieden Eher zufried  
## [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
```

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 z
```

```
## 2  Sehr zufrieden                               Eher unzufrieden Eher unz
```

```
## 3  Eher zufrieden                               Sehr zufrieden Eher unz
```

```
## 4  Eher zufrieden                               Sehr zufrieden Eher z
```

```
## 5  Eher zufrieden                               Eher zufrieden Eher z
```

```
## 6  Sehr zufrieden                               Eher zufrieden Eher z
```

```
## 7  Eher zufrieden                               Eher zufrieden Eher z
```

```
## 8  Eher zufrieden                               Eher zufrieden Eher z
```

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

Weitere Links

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

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.ssd</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")
```

- 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.

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

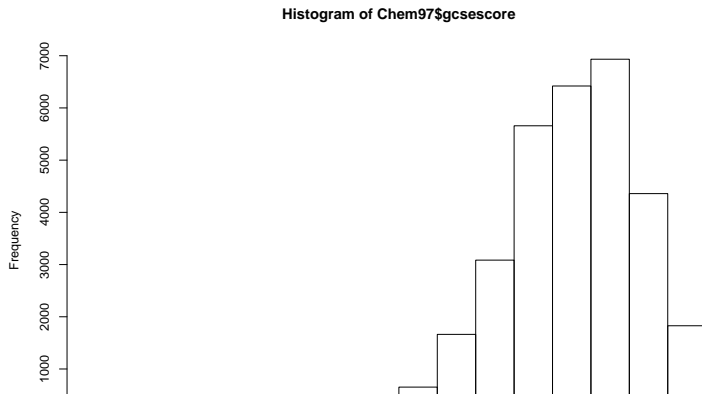
- [lea] Local Education Authority - a factor
- [school] School identifier - a factor
- [student] Student identifier - a factor
- [score] Point score on A-level Chemistry in 1997
- [gender] Student's gender
- [age] Age in month, centred at 222 months or 18.5 years
- [gcsecore] Average GCSE score of individual.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

Histogramm - Die Funktion hist()

Wir erstellen ein Histogramm der Variable gcsescore:

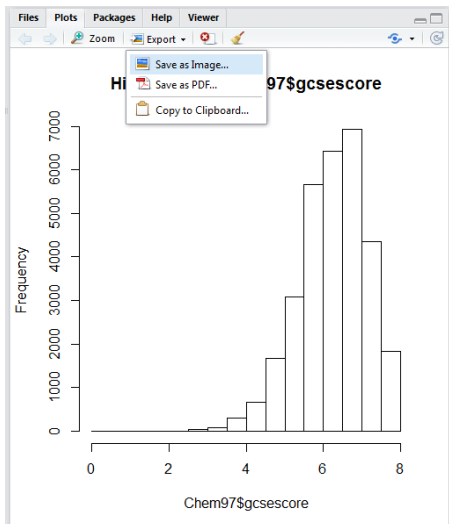
```
?hist
```

```
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()
```

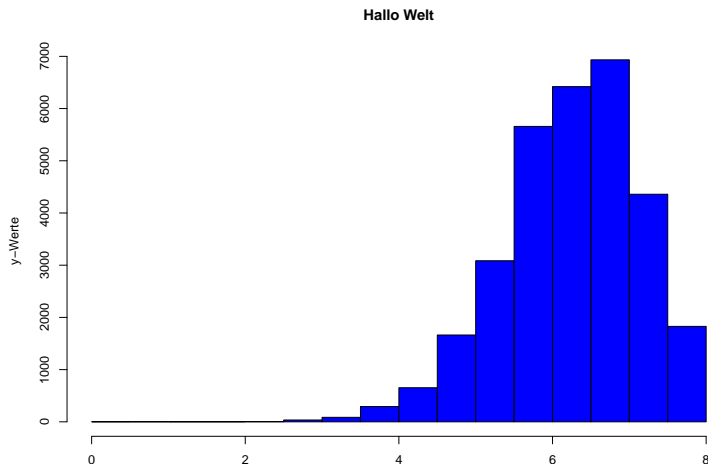
Histogramme

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

Histogramm

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



Barplot

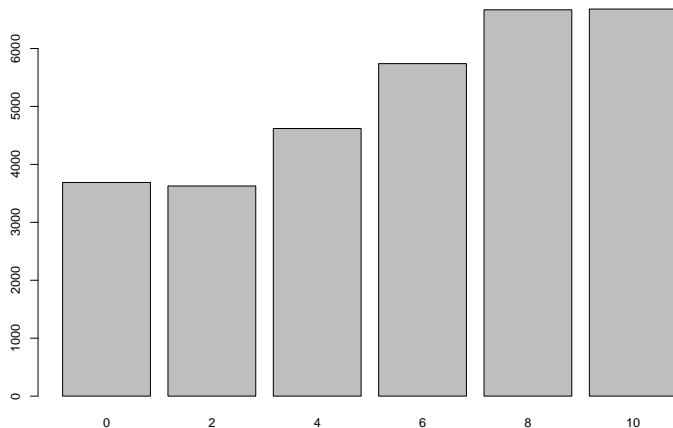
- Die Funktion `barplot()` erzeugt aus einer Häufigkeitstabelle einen Barplot
- Ist das übergebene Tabellen-Objekt zweidimensional wird ein bedingter Barplot erstellt

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

```
barplot(tabScore)
```

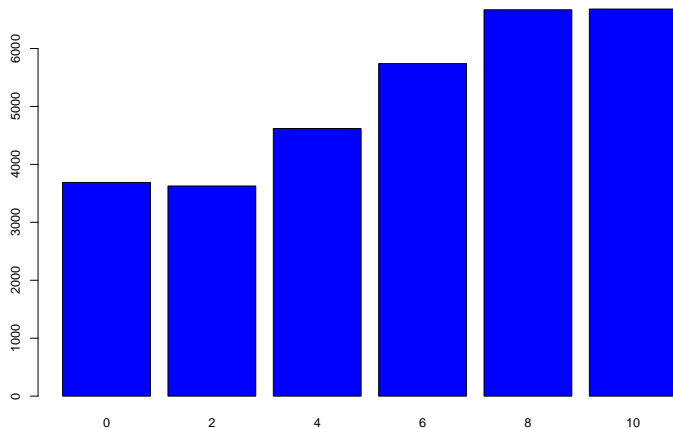

Barplots und barcharts

```
barplot(tabScore)
```



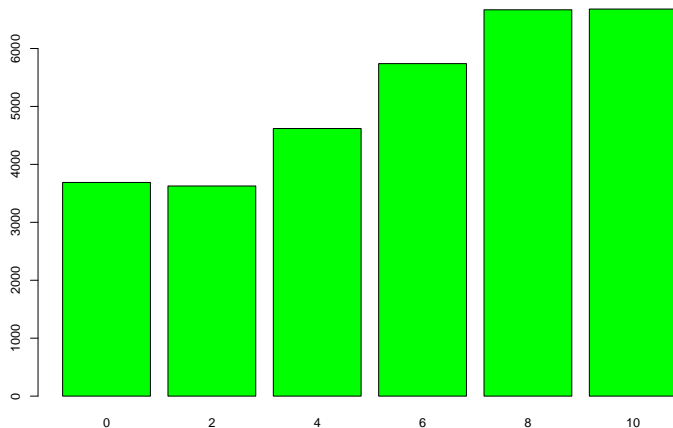
Mehr Farben:

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



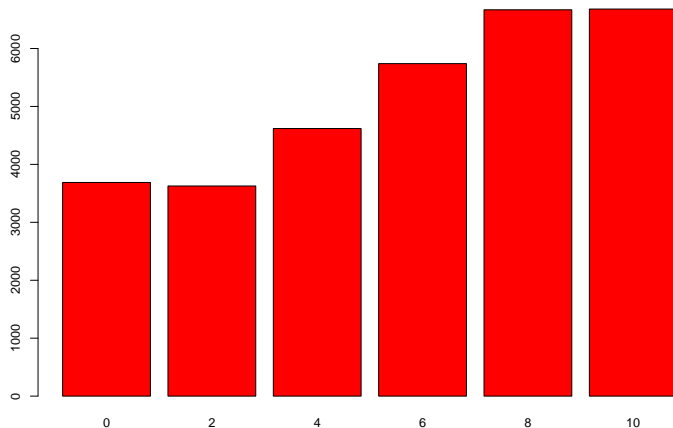
Grüne Farbe

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



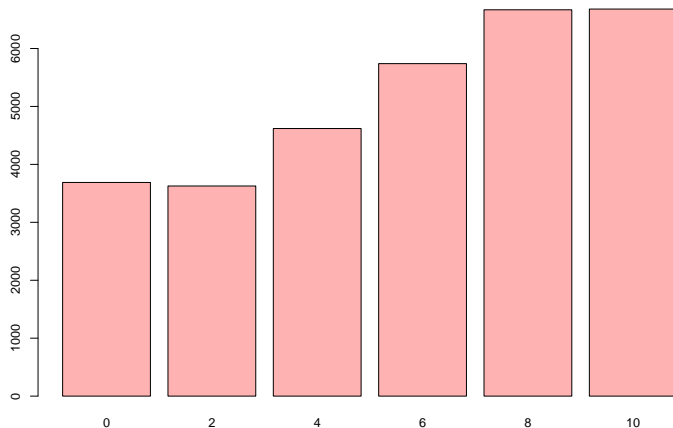
Rote Farbe

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



Transparent

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



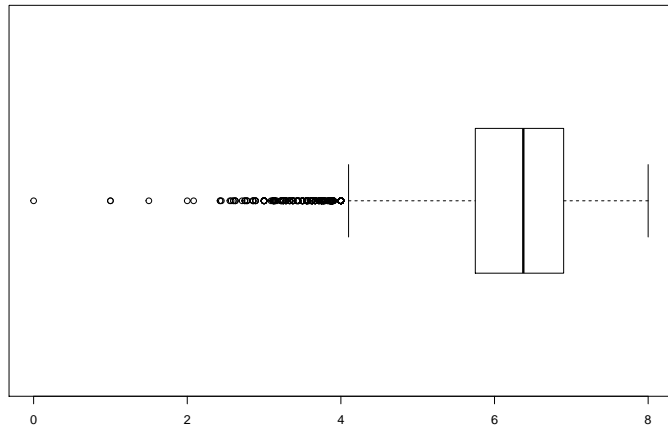
Boxplot

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

`?boxplot`

Horizontaler Boxplot

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

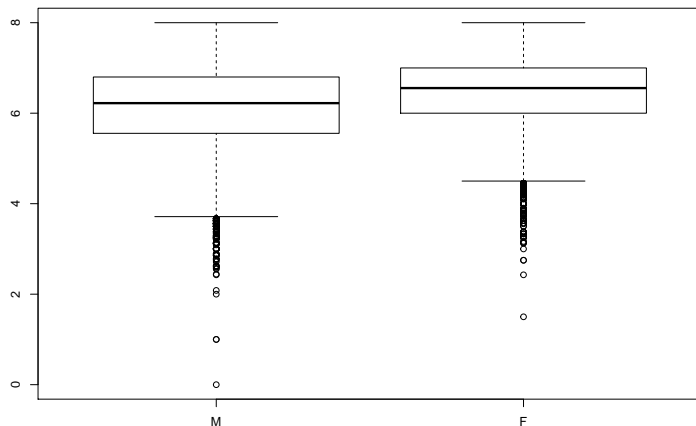


Gruppierte Boxplots

- 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 gruppierter Boxplot

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



Alternativen zu Boxplot

Violinplot

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

Beispieldaten erzeugen

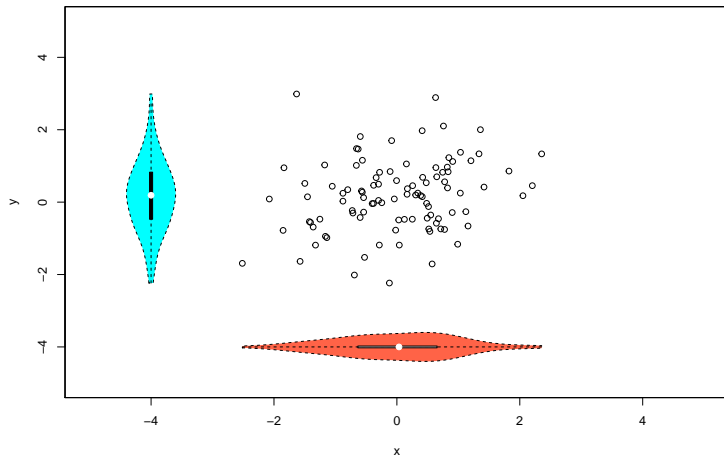
```
x <- rnorm(100)
```

```
y <- rnorm(100)
```

Die Bibliothek vioplot

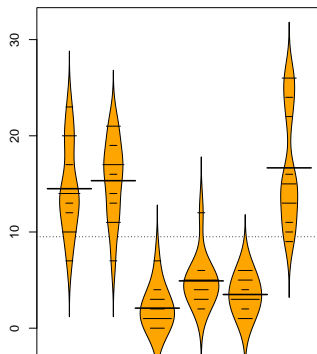
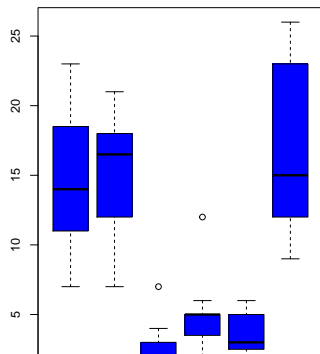
```
library(vioplot)
plot(x, y, xlim=c(-5,5), ylim=c(-5,5))
vioplot(x, col="tomato", horizontal=TRUE, at=-4,
        add=TRUE,lty=2, rectCol="gray")
vioplot(y, col="cyan", horizontal=FALSE, at=-4,
        add=TRUE,lty=2)
```

vioplot - Das Ergebnis



Alternativen zum Boxplot

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



Grafiken für bedingte, bi- und multivariate Verteilungen

Scatterplots

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

Datenanalyse

Streuungsmaße

In Basis R sind die wichtigsten Streuungsmaße enthalten:

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

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

```
## [1] 1.16249
```

```
sd(ab)
```

```
## [1] 1.078188
```

```
range(ab)
```

```
## [1] -2.927572 2.552941
```

Extremwerte

```
min(ab)
```

```
## [1] -2.927572
```

```
max(ab)
```

```
## [1] 2.552941
```

Fehlende Werte

- Sind NAs vorhanden muss dies der Funktion mitgeteilt werden

```
ab[10] <- NA  
var(ab)
```

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

Bei fehlenden Werten muss ein weiteres Argument mitgegeben werden:

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

```
## [1] 1.17203
```

Häufigkeiten und gruppierte Kennwerte

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

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

```
## x
##  1  2  3  4  5  6  7  8  9 10
##  9 11 10 12  5 13  8 10  8 14
```

Tabellieren - weiteres Beispiel

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

```
?table
```

```
table(x)
```

```
## x
```

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

```
## 9 11 10 12  5 13  8 10  8 14
```

```
table(x, musician)
```

```
##      musician
```

```
## x      no yes
```

```
## 1      3   6
```

```
## 2      2   9
```

```
## 3      5   5
```

```
## 4      6   6
```

Eine weitere Tabelle

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

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

Häufigkeitstabellen

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

Die Funktion `'prop.table()'`

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

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

Die Funktion `prop.table`

?prop.table

```
prop.table(table(esoph$agegp, esoph$alcgp), 1)
```

```
##
##           0-39g/day      40-79      80-119      120+
## 25-34 0.2666667 0.2666667 0.2000000 0.2666667
## 35-44 0.2666667 0.2666667 0.2666667 0.2000000
## 45-54 0.2500000 0.2500000 0.2500000 0.2500000
## 55-64 0.2500000 0.2500000 0.2500000 0.2500000
## 65-74 0.2666667 0.2000000 0.2666667 0.2666667
## 75+   0.2727273 0.3636364 0.1818182 0.1818182
```


Die aggregate Funktion

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

```
aggregate(state.x77,by=list(state.region),mean)
```

```
##           Group.1 Population    Income Illiteracy Life Exp
## 1      Northeast  5495.111  4570.222    1.000000  71.26444  4.
## 2           South  4208.125  4011.938    1.737500  69.70625 10.
## 3 North Central  4803.000  4611.083    0.700000  71.76667  5.
## 4           West  2915.308  4702.615    1.023077  71.23462  7.
##           Frost      Area
## 1 132.7778 18141.00
## 2  64.6250  54605.12
## 3 138.8333  62652.00
## 4 102.1538 134463.00
```

Beispieldatensatz - apply Funktion

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

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

```
## [1] 0.8631944 0.8914196 1.4921211 1.4176658
```

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

```
## [1] 2.5000000 0.7510758 0.2472249
```

Die Funktion `apply`

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

```
## [1] 0.02486923 1.33211279 1.72183151 5.15474598
```

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

```
## [1] 0.1576998 1.1541719 1.3121858 2.2704066
```

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

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.6907127 -0.3034952 0.6095059 -0.2651754
## [2,] 1.0000000  2.0000000 3.0000000  4.0000000
```

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

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

Argumente der Funktion `apply`

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

Die Funktion `tapply`

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

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

ApplyDat

```
##      Income Sex  
## 1 1335.796   2  
## 2 1388.897   2  
## 3 1505.359   1  
## 4 1589.819   1  
## 5 1431.924   2
```

Beispiel Funktion `tapply`

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

```
##           1           2
## 1547.589 1385.539
```

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

```
## $`1`
## [1] 1505.359 1589.819
##
## $`2`
## [1] 1335.796 1388.897 1431.924
```

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

Die Daten laden

```
library(foreign)
dat <- read.dta("https://github.com/Japhilko/RSocialScience/bl
```

Eine Kreuztabelle erstellen

```
Beruf_Gefordert <- dat$a11c109a
Beruf_Anerkannt <- dat$a11c111a
```

```
table(Beruf_Gefordert, Beruf_Anerkannt)
```

```
##                Beruf_Anerkannt
## Beruf_Gefordert  Missing by design Ja Nein Weiß nicht
##  Missing by design          93  0    0              0
##    Ja                    0  7    0              0
##    Nein                   0  0    0              0
##    Weiß nicht            0  0    0              0
```

Eine Dreidimensionale Kreuztabelle - Array

```
Geschlecht <- dat$a11d054a
tab3 <- table(Beruf_Gefordert,Beruf_Anerkannt,Geschlecht)
tab3
```

```
## , , Geschlecht = Männlich
```

```
##
## Beruf_Anerkannt
```

```
## Beruf_Gefordert    Missing by design Ja Nein Weiß nicht
## Missing by design      41  0    0          0
## Ja                    0  2    0          0
## Nein                   0  0    0          0
## Weiß nicht            0  0    0          0
```

```
##
## , , Geschlecht = Weiblich
```

```
##
## Beruf_Anerkannt
```

Indizieren eines Arrays

- nun muss man mit zwei Kommas arbeiten beim Indizieren

```
tab3[, , 1]
```

```
##                Beruf_Anerkannt
## Beruf_Gefordert  Missing by design Ja Nein Weiß nicht
##  Missing by design          41  0    0              0
##    Ja                    0  2    0              0
##    Nein                   0  0    0              0
##    Weiß nicht            0  0    0              0
```

Edgar Anderson's Iris Daten

```
data(iris)
```

```
head(iris)
```

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

petal length and width - Blütenblatt Länge und Breite

sepal length and width - Kelchblatt Länge und Breite

- [Wikipedia Artikel zum IRIS Datensatz](#)

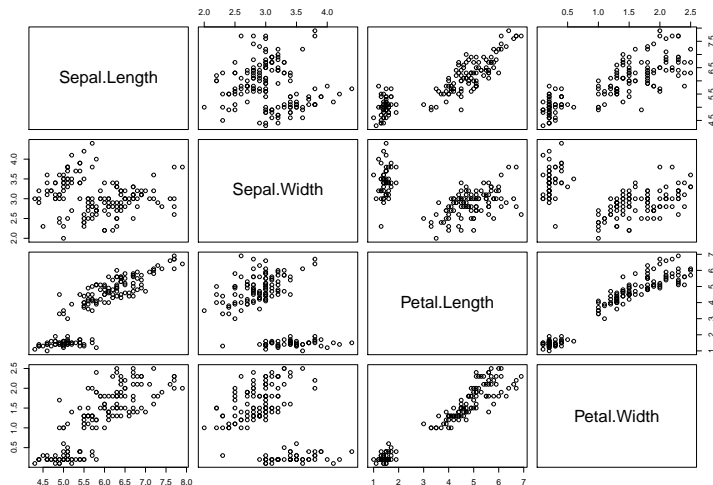
Zusammenhang zwischen stetigen Variablen

```
# Pearson Korrelationskoeffizient  
cor(iris$Sepal.Length,iris$Petal.Length)  
  
## [1] 0.8717538
```

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

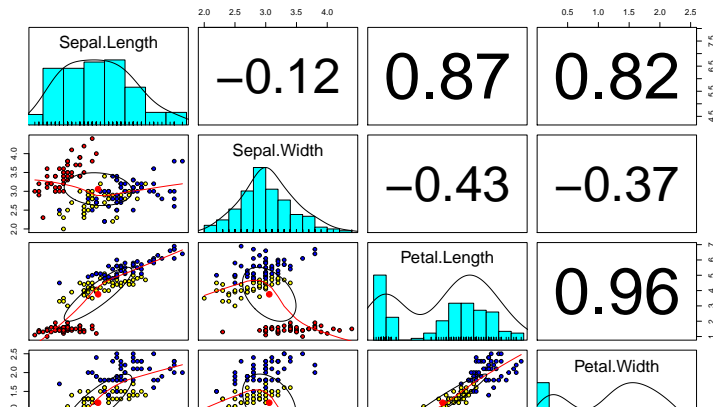
Zusammenhang zwischen mehreren Variablen

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



Zusammenhang zwischen mehreren Variablen

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



Verschiedene Korrelationskoeffizienten

```
# Pearson Korrelationskoeffizient
```

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

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

```
# Kendall's tau (Rangkorrelation)
```

```
cor(iris[,1:4], method = "kendall")
```

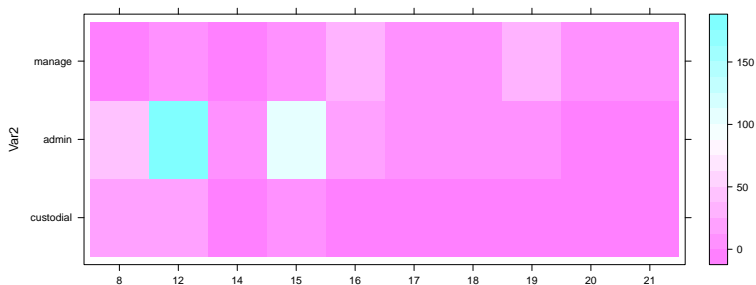
```
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length      1.00000000 -0.07699679    0.7185159    0.65530856
## Sepal.Width       -0.07699679    1.00000000   -0.1859944   -0.15712566
## Petal.Length       0.71851593 -0.18599442    1.0000000    0.8068907
## Petal.Width   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

Zusammenhang zwischen kategorialen Variablen

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

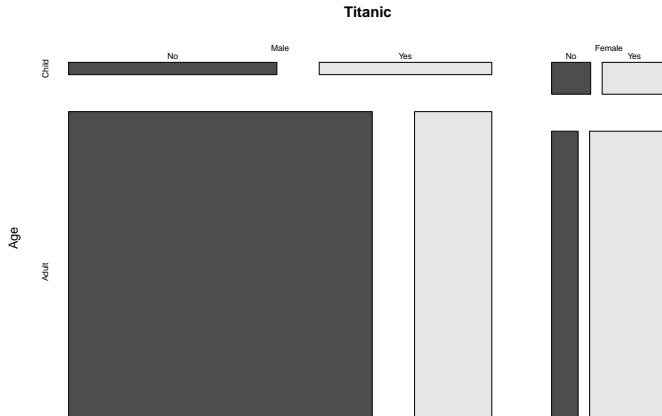
Levelplot

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



Visualisierung von Zusammenhängen zwischen kategorialen Variablen

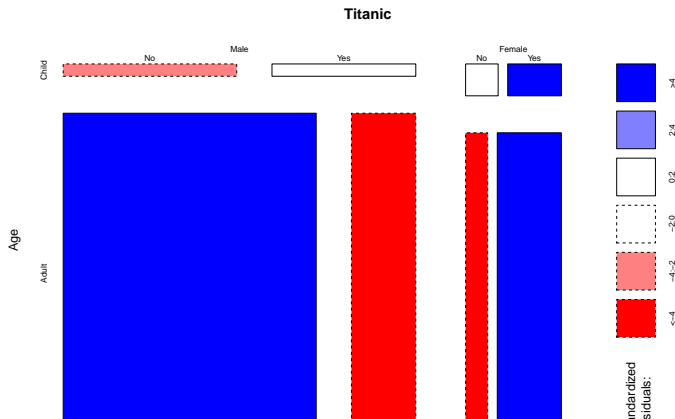
```
mosaicplot(~ Sex + Age + Survived,
            data = Titanic, color = TRUE)
```



Shading

Flächen werden entsprechend der Residuen eingefärbt:

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



Literatur zu Zusammenhangsmaßen

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

Sachs - Angewandte Statistik mit R

Das lattice Paket

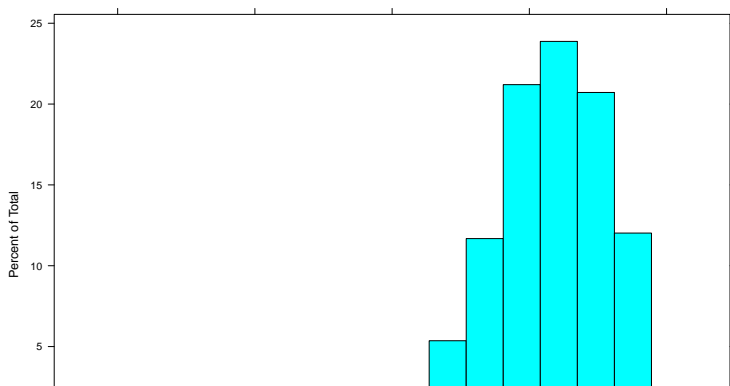
Das lattice-Paket

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

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

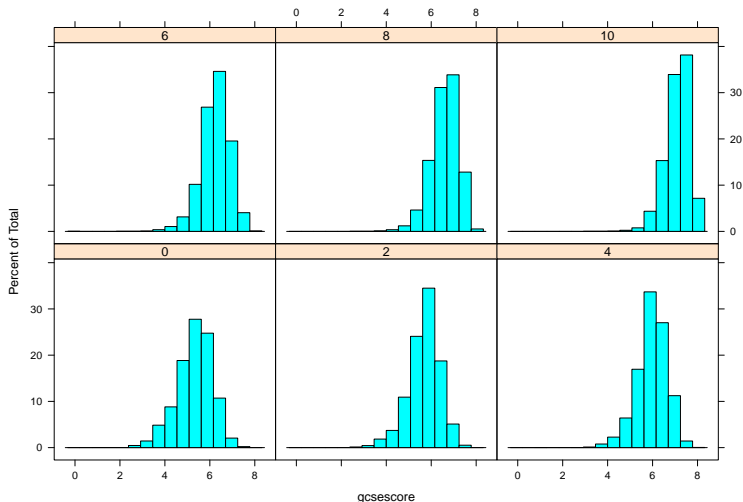
Histogramm mit Lattice

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



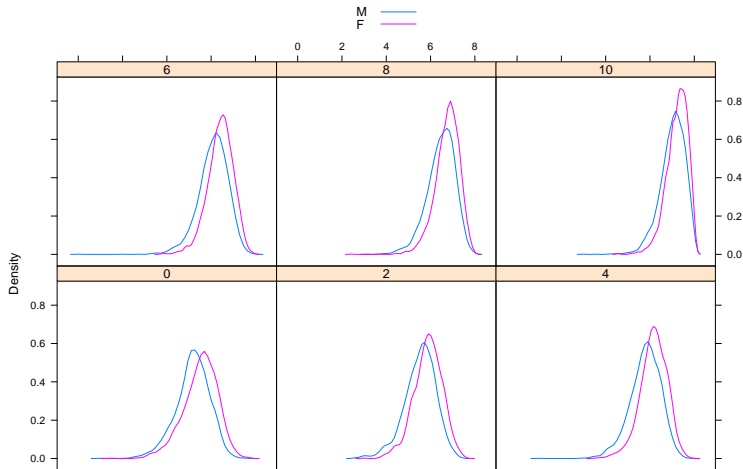
Histogramm mit Lattice

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



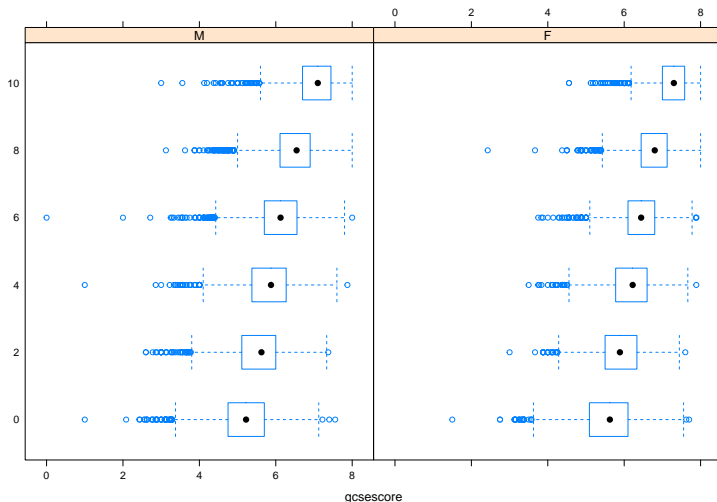
Die Dichte mit Lattice zeichnen

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



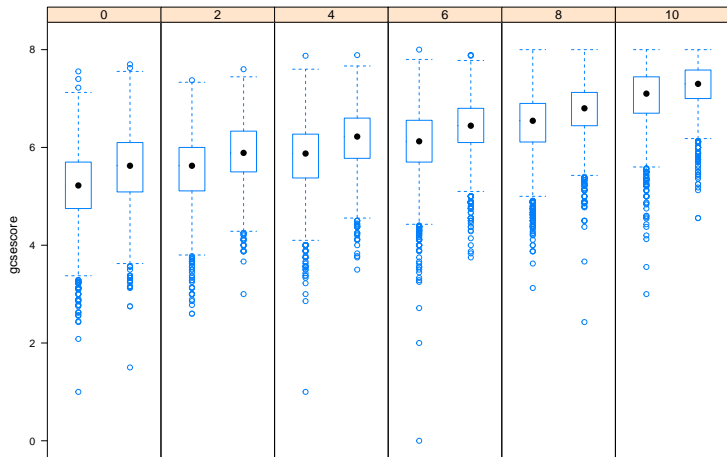
Boxplot mit Lattice zeichnen

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



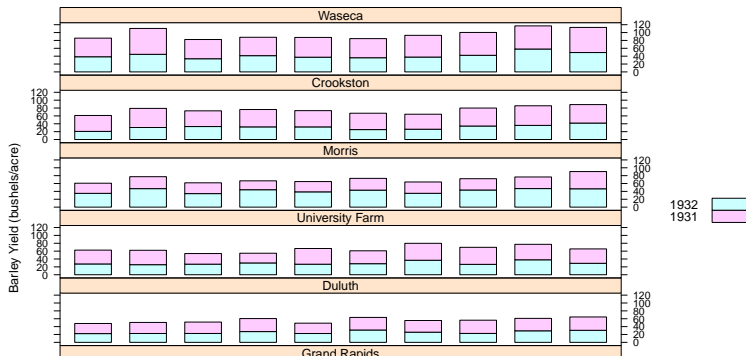
Boxplot mit Lattice zeichnen

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



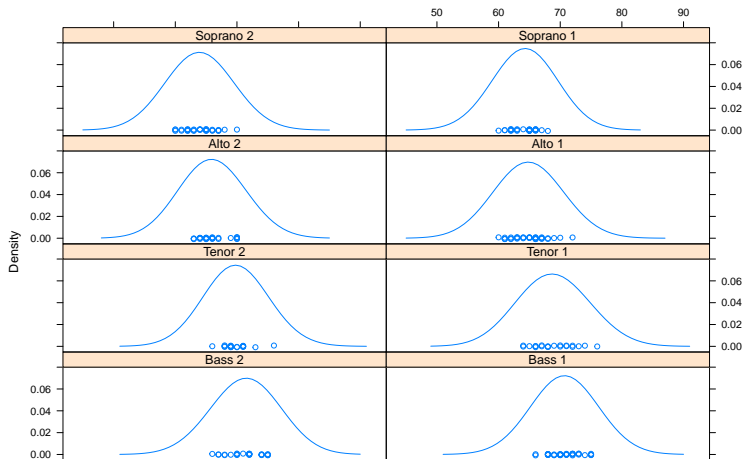
Univariate Plots

```
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)))
```



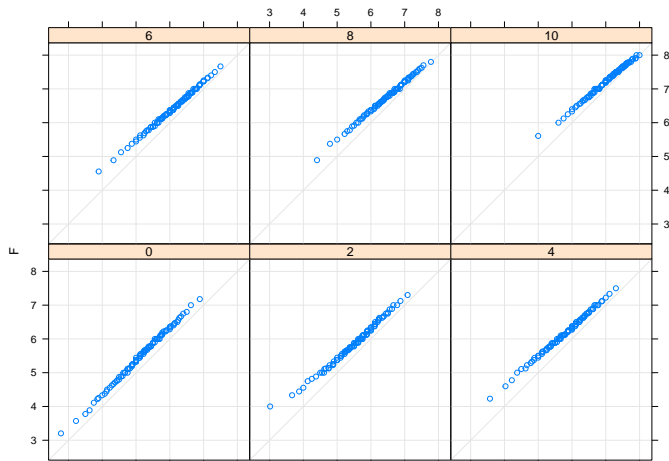
Densityplot

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



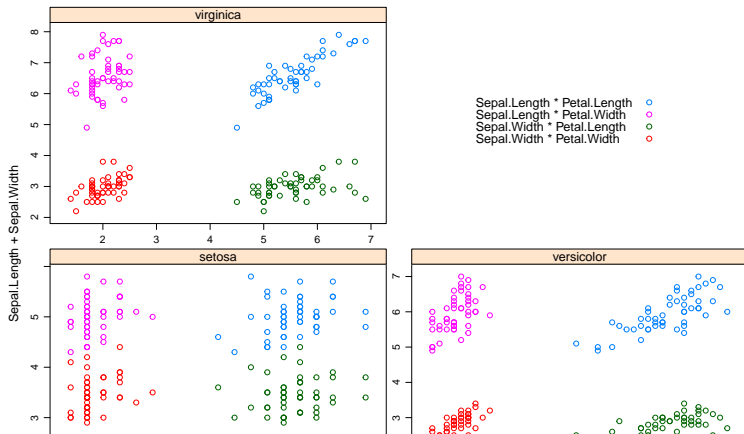
Bivariate Plots

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



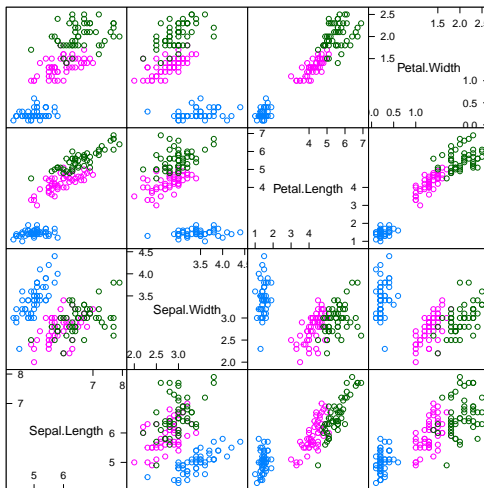
xyplot

```
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)
```



Scatter Plot Matrix

parallelplot

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

