

WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 10. April 2018

[Module und Studiengänge](#)

[Zur Vorlesung](#)

[Zur Übung](#)

[Zur Modulprüfung](#)

[Literatur zur Lehrveranstaltung](#)

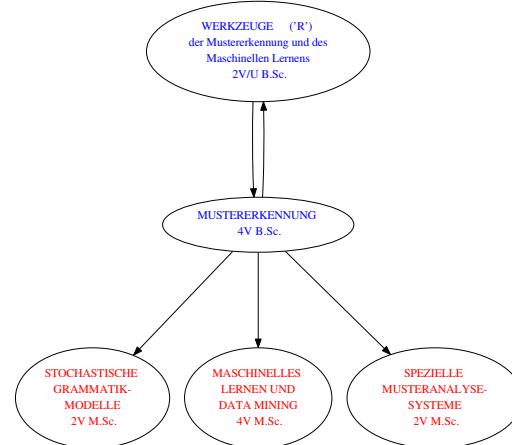
Professur für Musteranalyse

Lehrangebot Wintersemester & Sommersemester

Lehrbereich

Informatik
WP-Bereich
Intelligente Systeme
Vertiefung Künstliche Intelligenz und Mustererkennung
Statistische Musteranalyse

Inhaltliche Abhängigkeiten



Meine Lehrveranstaltungen für ...

Informatiker & Bioinformatiker & Informatikerinnen & Bioinformatikerinnen

	SOMMERSEMESTER	WINTERSEMESTER
ASQ und EF Inf.	<i>Intelligente Systeme</i> 4V	Literaturarbeit und Präsentation ^{ASQ} 2S
Bachelor	Mustererkennung 4V Werkzeuge ME/ML 2Ü	<i>Strukturiertes Programmieren</i> 4V+2Ü
Master	Stochastische Grammatikmodelle 2V Biometriesysteme (Seminar) 2S	Maschinelles Lernen und Datamining 4V Spezielle Musteranalyse-systeme 2V

Wie studiere ich MUSTERANALYSE ?

Studiengänge: Informatik · Bioinformatik · Angewandte Informatik

Bachelor

Mustererkennung^{6LP}



WeRkzeuge ME/ML^{3LP}



Vertiefungsangebote

auf Antrag beim Prüfungsaamt:

Maschinelles Lernen^{6LP}



Master

Maschinelles Lernen^{6LP}



Stochast. Grammatik^{3LP}



Musteranalysesysteme^{3LP}



Nivellierungsmodule

auf Antrag beim Prüfungsaamt:

Mustererkennung^{6LP}



WeRkzeuge ME/ML^{3LP}



Mustererkennung

Vorlesung der Bachelor/Master-Studiengänge · 4V · 6 LP

Modultitel (deutsch)	Mustererkennung
Modultitel (englisch)	Pattern Recognition
Modulnummer	FMI-IN0036
Art des Moduls (Pflicht-, Wahlpflicht- oder Wahlfmodul)	Wahlpflichtmodul (INT) für den B.Sc. Informatik Wahlpflichtmodul (INT) für den B.Sc. Angewandte Informatik Wahlpflichtmodul (Wahlpflichtbereich 2) für den B.Sc. Bioinformatik Wahlpflichtmodul (INT) für den M.Sc. Informatik (auf Antrag) Wahlpflichtmodul für den M.Sc. Bioinformatik (Bereich Informatik) Pflichtmodul für das Anwendungsfach Computational Neuroscience zum B.Sc. Angewandte Informatik Wahlpflichtmodul für das Lehramt Informatik
Modul-Verantwortlicher	Ernst Günther Schukat-Talamazzini
Leistungspunkte (ECTS credits)	6
Arbeitsaufwand (work load) in:	180 Std. 60 Std. 120 Std.
- Präsenzstunden	
- Selbststudium (einschl. Prüfungsvorbereitung)	
Lehrform (SWS)	3 V + 1 Ü
Häufigkeit des Angebots (Modulturnus)	jährlich im Sommersemester
Dauer des Moduls	1 Semester
Voraussetzung für die Zulassung zum Modul	
Empfohlene Vorkenntnisse für das Modul	- FMI-IN0070 (Grundlagen der Modellierung und Programmierung) oder FMI-IN0040 (Grundlagen der Modellierung und Programmierung (Grundteil)) oder FMI-IN0025 (Strukturiertes Programmieren für Bioinformatiker) - FMI-IN0001 (Algorithmen und Datenstrukturen) - FMI-MA0007 (Einführung in die Wahrscheinlichkeitstheorie)
Voraussetzung für die Zulassung zur Modulprüfung	Bearbeitung der Übungsaufgaben Mindestens 50% der erzielbaren Punkte erreicht
Voraussetzung für die Vergabe von Leistungspunkten (Prüfungsform)	Klausur (120min) oder mündliche Prüfung (30min) zur Vorlesung Studienleistungsergänzung: Erfolgsmetriken. Abgestufte (Prüfungs-)Anforderungen berücksichtigen das von Bachelor- und Masterstudierenden jeweils erreichbare Leistungsniveau.
Inhalte	Einführung in die Methoden der Mustererkennung zur maschinellen Modellierung und Simulation komplexer Informationsverarbeitungsprozesse. Besonders sie insbesondere bei der Wahlermung und Auswertung visueller, akustischer oder taktiler Sinnesreize durch den Menschen aufreten. Diskretisierung/Filtrierung/Normierung; Merkmalsauswahl und Merkmalstransformation; statistische, diskriminative und nichtparametrische Klassifikatoren; unüberwachtes Lernen; Zeitreihen

B.Sc. Informatik

WP-Bereich Int.Syst.

B.Sc. Bioinform.

WP-Bereich

B.Sc. Ang.Inform.

WP-Bereich Int.Syst.

Pflicht im Anw.fach CNS

M.Sc. Informatik

WP-Bereich Int.Syst.

M.Sc. Bioinform.

WP-Bereich Int.Syst.

LG Informatik

FS 6–9

Werkzeuge Mustererkennung & Maschinelles Lernen

Vorlesung der Bachelor/Master-Studiengänge · 2V/P · 3 LP

Modultitel (deutsch)	Werkzeuge der Mustererkennung und des Maschinellen Lernens
Modultitel (englisch)	Tools for Pattern Recognition and Machine Learning
Modulnummer	FMI-IN0086
Art des Moduls (Pflicht-, Wahlpflicht- oder Wahlfmodul)	Wahlpflichtmodul (INT) für B.Sc. Informatik Wahlpflichtmodul (INT) für B.Sc. Angewandte Informatik Wahlpflichtmodul (KIME, INT) für den M.Sc. Informatik (auf Antrag) Wahlpflichtmodul für den B.Sc. Bioinformatik (Bereich Informatik)
Modul-Verantwortlicher	Ernst Günther Schukat-Talamazzini
Leistungspunkte (ECTS credits)	3
Arbeitsaufwand (work load) in:	90 Std. 30 Std. 60 Std.
- Präsenzstunden	
- Selbststudium (einschl. Prüfungsvorbereitung)	
Lehrform (SWS)	2V (mit Übung)
Häufigkeit des Angebots (Modulturnus)	jedes Sommersemester
Dauer des Moduls	1 Semester
Voraussetzung für die Zulassung zum Modul	Keine
Empfohlene Vorkenntnisse für das Modul	FMI-IN0036 (Mustererkennung) sollte gleichzeitig belegt werden
Voraussetzung für die Zulassung zur Modulprüfung	50% der erreichbaren Punkte aus den Übungsaufgaben
Voraussetzung für die Vergabe von Leistungspunkten (Prüfungsform)	Mündliche Prüfung oder Klausur
Inhalte	Aufgabenstellungen aus den Bereichen Mustererkennung, Maschinelles Lernen, Datamining und ihre Bearbeitung mit geeigneten Softwarewerkzeugen: Klassifikation, Vorhersage, Clustering, Transformation, Visualisierung, Zeitreihen, Spektraldarstellung, Wahrscheinlichkeitsmodelle
(Qualifikations-)Ziele	- Fähigkeiten im praktischen Umgang mit Entwicklungswerkzeugen für maschinelles Lernen in Musteranalyse und Datamining - Grundlegende Kenntnisse über den Aufbau von Softwaresystemen und Programmierparadigmen für die maschinelle Datenanalyse - Kompetenzen in Datenanalyse, Versuchsplanung, Konfiguration von ML-Systemen

Maschinelles Lernen & Data Mining

Vorlesung der Master-Studiengänge · 4V · 6 LP

Modultitel (deutsch)	Maschinelles Lernen und Datamining
Modultitel (englisch)	Machine Learning and Datamining
Modulnummer	FMI-IN0034
Art des Moduls (Pflicht-, Wahlpflicht- oder Wahlfmodul)	Wahlpflichtmodul (KIME, INT) für den M.Sc. Informatik Wahlpflichtmodul (INT) für den B.Sc. Informatik (zusätzliches Lehrangebot) Wahlpflichtmodul für den M.Sc. Bioinformatik (Bereich Informatik) Wahlpflichtmodul (INF) für den M.Sc. Computational Science Wahlpflichtmodul für das Lehramt Informatik
Modul-Verantwortlicher	Ernst Günther Schukat-Talamazzini
Leistungspunkte (ECTS credits)	6
Arbeitsaufwand (work load) in:	180 Std. 60 Std. 120 Std.
- Präsenzstunden	
- Selbststudium (einschl. Prüfungsvorbereitung)	
Lehrform (SWS)	4V (mit Projektanteil)
Häufigkeit des Angebots (Modulturnus)	jährlich im Wintersemester
Dauer des Moduls	1 Semester
Voraussetzung für die Zulassung zum Modul	Keine
Empfohlene Vorkenntnisse für das Modul	FMI-IN0036 (Mustererkennung)
Voraussetzung für die Zulassung zur Modulprüfung	Keine
Voraussetzung für die Vergabe von Leistungspunkten (Prüfungsform)	Klausur (120min) oder mündliche Prüfung (30min) zur Vorlesung
Inhalte	Strukturaufdeckung, Klassifizierung oder Entwicklungsvorschläge aus großen Datenflächen (Finanzprozesse, Handel und Transport, med./biol. Datenstätten, Klimamesswerte, elektronische Dokumente, Fertigungsaufbereitung) – Vorlesungsthemen sind u.a.: Skalentypen; Visualisierung hochdimensionaler Daten (PCA, MDS, ICA); überwachte Lernverfahren (Versionenraum, Entscheidungsbaum, lineare/logistische Modelle); unüberwachte Lernverfahren (hierarchisch, (fuzzy) K-means, spektral); Graphische Modelle (Bayesnetze, Markovketten, Induktion und Inferenz)
(Qualifikations-)Ziele	Tieffrequente Fachkenntnisse des Gebiets Maschinelles Lernen Fähigkeit zur Analyse, Design und Realisierung von ML-Systemen Flächendeckende Übersicht aktueller Techniken des Datamining

M.Sc. Informatik

WP-Bereich Int.Syst.
Schwerpunkt KI/ME

M.Sc. Bioinform.

WP-Bereich

M.Sc. Comp.Science

WP-Bereich

B.Sc. Informatik

Zusatzangebot

LG Informatik

FS 6–9

Stochastische Grammatikmodelle

Vorlesung der Master-Studiengänge · 2V · 3 LP

Modul FMI-IN0146 Stochastische Grammatikmodelle	
Modulnummer/-code	FMI-IN0146
Modultitel (deutsch)	Stochastische Grammatikmodelle
Modultitel (englisch)	Stochastic Grammars
Modulverantwortlicher	Ernst Günter Schukat-Talamazzini
Voraussetzungen für Zulassung zum Modul	keine
Empfohlene bzw. erwartete Vorkenntnisse	keine
Art des Moduls (Pflicht-, Wahlpflicht- oder Wahlmodul)	Wahlpflichtmodul (KIME, INT) für den M.Sc. Informatik Wahlpflichtmodul für den M.Sc. Bioinformatik (Bereich bioinformatisch relevante Informatik) Wahlpflichtmodul für den M.Sc. Mathematik (Nebenfach Informatik) Wahlpflichtmodul (INF) für den M.Sc. Computational Science
Häufigkeit des Angebots (Zyklus)	jedes 2. Semester (ab Sommersemester)
Dauer des Moduls	1 Semester
Zusammensetzung des Moduls / Lehrformen (VL, Ü, S, Praktikum)	2V
Leistungspunkte (ECTS credits)	3LP
Arbeitsaufwand (work load)	90h
- Präsenzstunden	30h
- Selbststudium (einschl. Prüfungsvorbereitungen)	60h
Inhalte	Grammatische Modellierung von Zeichenfolgen natürlicher („Texte“) und künstlicher (z.B. Nukleotid- oder Aminosäuresequenzen) Sprachen. Vorlesungsthemen sind u.a.: <ul style="list-style-type: none">• Schwach kontextfreie Grammatiken (IG, TAG, HG, CG)• Information/Kompression• robuste Häufigkeitsschätzung (Bayes, Good-Turing, Zipf)• N-Gramme, Interpolation, Maximum-Entropiestochastische

Spezielle Musteranalysesysteme

Vorlesung der Master-Studiengänge · 2V · 3 LP

Modultitel (deutsch)	Spezielle Musteranalysesysteme
Modultitel (englisch)	Pattern Analysis Systems
Modulnummer	FMI-IN0054 02.12.09
Art des Moduls (Pflicht-, Wahlpflicht- oder Wahlmodul)	Wahlpflichtmodul (INT) für den M.Sc. Informatik Wahlpflichtmodul für den M.Sc. Bioinformatik (Bereich Informatik)
Modul-Verantwortlicher	Ernst Günter Schukat-Talamazzini
Leistungspunkte (ECTS credits)	3
Arbeitsaufwand (work load) in:	90 Std. - Präsenzstunden - Selbststudium (einschl. Prüfungsvorbereitung)
Lehrform (SWS)	2V (mit Projektanteil)
Häufigkeit des Angebots (Modultypus)	jährlich im Sommersemester
Dauer des Moduls	1 Semester
Voraussetzung für die Zulassung zum Modul	keine
Empfohlene Vorkenntnisse für das Modul	- FMI-IN0036 (Mustererkennung) - Vorkenntnisse aus den Bereichen Künstliche Intelligenz und Digitale Bildverarbeitung
Voraussetzung für die Vergabe von Leistungspunkten (Prüfungsform)	mündliche Prüfung (30min) zur Vorlesung oder Ausarbeitung/Präsentation zu einer Projektaufgabe
Inhalte	- Komplexe Musteranalyseaufgaben mit longitudinalen Daten (Sprach- und Sprecherkennung, Handschrifterkennung, DNA-Motive, Musikretrieval) - Geeignete Lernverfahren (z.B. Hidden Markov Modelle; siehe Webseite zum Kurs für Detailinformationen), unterstützende Werkzeuge, Vorverarbeitung und Etikettierung der Lerndaten und syntaktische Modellierungsverfahren am Beispiel einer oder mehrerer ausgewählter Aufgabenstellungen
(Qualifikations-)Ziele	- Vertiefte Kenntnis der Methoden syntaktischer Musteranalyse - Kompetenzen der Analyse, des Designs und der Realisierung von Musteranalysesystemen realistischer Größenordnung - Fertigkeiten der Nutzung ausgewählter Softwarewerkzeuge der syntaktischen Musteranalyse

Module und Studiengänge

Zur Vorlesung

Zur Übung

Zur Modulprüfung

Literatur zur Lehrveranstaltung

M.Sc. Informatik

WP-Bereich Int.Syst.

Schwerpunkt KI/ME

M.Sc. Bioinform.

WP-Bereich

M.Sc. Comp.Science

WP-Bereich

Vorlesungsstoff

Inhaltliche Strukturierung

1. Die Programmiersprache

Sprachumfang, Installation, Sitzung, Elementargrafik

2. Rechnen in : die wichtigen Klassen

Vektor, Matrix, Liste, Dataframe, Faktor, Kontrolle, Vektorisierung, Funktionen, -Klassen

3. Datensätze & Grafikfunktionen in

Zeichenkette, Datensatz, E/A, Paket · Grafik: 1D, 2D, 3D, HD

4. Modellieren in

Statistikfunktionen, Verteilungen, Formula-Objekte, Vorhersagemodelle, Optimierung

5. Ausgewählte Werkzeuge für ME/ML/DM

Signalverarbeitung, Transformation, Regression, Klassifikation, Clustering

Vorlesung

Sitzungsablauf und Vorlesungsfolien

Semesterstart

(*Sitzungen 1–3*)

90 Minuten Vorlesungsfolien

Semesterhauptteil

(*Sitzungen 4–14*)

45 Minuten Übungsaufgaben

45 Minuten Vorlesungsfolien

- Die Vorlesungs*inhalte* sind **hierarchisch** gegliedert.
(wenn man Glück hat)
- Die Vorlesung selbst besitzt svst. eine streng **lineare** Abfolge.
- Ein Vorlesungsskript auch,
aber Sie können es in beliebiger Reihenfolge lesen.
- Das Folienskript folgt der inhaltlichen Hierarchie ...
 - ➡ Einfaches neben Schwierigem
 - ➡ Standards neben Exoten

Vorlesung

Nutzung der Folienpräsentation

- Die Folien sollen vom Mitschreiben während der Vorlesung entlasten.
- Das Mitschreiben wird dadurch nicht überflüssig.
- Die Folien sind kein Lehrbuch.
- Die Folien sind daher im allgemeinen nur mit den Erläuterungen während der Vorlesung und entsprechenden eigenen Notizen verständlich.

Module und Studiengänge

[Zur Vorlesung](#)

[Zur Übung](#)

[Zur Modulprüfung](#)

Literatur zur Lehrveranstaltung

Übungsaufgaben

Bearbeitung der Aufgabenblätter

URL: <http://www.minet.uni-jena.de/fakultaet/schukat/WMM/SS18/node2.html>

Aufgabenblätter

10 Übungsblätter (Woche #3 bis Woche #12)
 2–3 Programmieraufgaben je Blatt (WWW)
 Ü-Blatt/PDF & Datensätze, Infotexte, Codeschnipsel ...

Bearbeitung

Teamarbeit oder Arbeitsteilung
 Verweise auf Skript „*Mustererkennung*“
 Rückfragen per e-Mail
 (2 Mitglieder/Gruppe)
 (Kapitel/Seitenzahl)
 (*reply* ~ ALLE)

Abgabe der Lösung

Programmcode
 Schriftliche Antworten
 Grafikausgaben, Tabellen
 (Textdatei)
 e-Mail mit Anhängen oder Archiv
 (Textdatei oder PDF)
 (PDF, Text)

Übungsaufgaben

Lösen der Hausaufgaben in Arbeitsgruppen

Arbeitsgruppen

Hausaufgabenblätter werden in Gruppenarbeit gelöst.
Jede Arbeitsgruppe besteht aus **genau 2** Mitgliedern.
(*Ausnahmen nur nach persönlicher Absprache*)

Anmeldung per e-Mail

Jede Gruppe meldet sich bis **Montagabend (16.4.2018)** an.
Gruppenname plus (Vor/Zuname/Mail) je Mitglied.

Kostenloses Speed-Dating

„Singles“ melden sich bitte ebenfalls bei mir (s.o.).
Sie werden dann im *round-robin*-Verfahren verkuppelt.

Übungsaufgaben

Workflow für die Bearbeitung der Übungsserien

Ausgabe

Woche M

Die neue Aufgabenstellung wird *freitags* im Internet veröffentlicht.

Abgabe

Woche M + 1

Die Arbeitsgruppen liefern ihre Lösungen ab.
(≤ 1 Papierstoß, ≤ 1 Archiv, ≤ 1 e-Mail, bitte!)
Deadline ist **Sonntag 23:59 Uhr**

Rückgabe & Besprechung

Woche M + 2

Die Aufgaben und ihre Lösung werden *im Falle öffentlichen Interesses* in der Übungsveranstaltung besprochen.

Erstes Übungsblatt

Ausgabe am Freitag, den 20.4.2018

Module und Studiengänge

Zur Vorlesung

Zur Übung

Zur Modulprüfung

Literatur zur Lehrveranstaltung

Prüfungsgegenstand

Bearbeitung der Übungsaufgabenblätter
Abgabe der Lösungen (gruppenweise)
Summe erzielter Punkte

Prüfungstermine

(*nur in Einzel- und Ausnahmefällen*)

Prüfungsstoff

Programmierung in 'R'

Anmeldung und Zulassung

zur Teilnahme **und** zur Prüfung
Deadline: **Dienstag, 22.05.2018**



Module und Studiengänge

Zur Vorlesung

Zur Übung

Zur Modulprüfung

Literatur zur Lehrveranstaltung

Programmieren in 'R'

Empfohlene Bücher zur Vorlesung

-  **W.N. Venables and B.D. Ripley.**
Modern Applied Statistics with S.
Springer, 2002.
-  **U. Ligges.**
Programmieren mit 'R'.
Springer, 2004.
-  **Norman Matloff.**
Art of R Programming.
No Starch Press, 2011.
-  **Jürgen Groß.**
Grundlegende Statistik mit R.
Vieweg, Wiesbaden, 2010.
-  **Dirk Eddelbuettel.**
Seamless R and C++ Integration with Rcpp.
Number 64 in Use R! Springer, 2013.

Daten- und Musteranalyse in 'R'

Empfohlene Bücher zur Vorlesung

-  **Brian Everitt and Torsten Hothorn.**
An Introduction to Applied Multivariate Analysis with R.
Use R. Springer, 2011.
-  **Paul S.P. Cowpertwait and Andrew V. Metcalfe.**
Introductory Time Series with R.
Use R! Springer, 2009.
-  **Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang.**
Principles and Theory for Data Mining and Machine Learning.
Springer Series in Statistics. Springer, 2009.
-  **Graham Williams.**
Data Mining with Rattle and R.
Use R. Springer, 2011.
-  **Randall Pruim.**
Foundations and Applications of Statistics: An Introduction Using R, volume 13 of Pure and Applied Undergraduate Texts.
American Mathematical Society, 2011.

Weitere Anwendungen für 'R'

Empfohlene Bücher zur Vorlesung

-  **Søren Højsgaard, David Edwards, and Steffen Lauritzen.**
Graphical Models with R.
Use R! Springer, 2012.
-  **Jim Albert.**
Bayesian Computation with R.
Use R! Springer, 2009.
-  **F. Hahne, W. Huber, R. Gentleman, and S. Falcon.**
Bioconductor Case Studies.
Use R. Springer, 2008.
-  **Giovanni Petris, Sonia Petrone, and Patrizia Campagnoli.**
Dynamic Linear Models with R.
Use R! Springer, 2009.
-  **Eric D. Kolaczyk and Gábor Csárdi.**
Statistical Analysis of Network Data with R.
Use R! Springer, 2014.

Maschinelles Lernen

Empfohlene Bücher zur Vorlesung

-  **G. James, D. Witten, T. Hastie, and R. Tibshirani.**
An Introduction to Statistical Learning.
Number 103 in Springer Texts in Statistics. Springer, 2013.
-  Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.
Hardcover 60 EUR.
-  T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer, 2001.
-  R.O. Duda, P.E. Hart, and D.G. Stork.
Pattern Classification.
Wiley Interscience, 2001.
-  S. Furui.
Digital Speech Processing, Synthesis, and Recognition.
Marcel Dekker, New York, 2001.

Data Mining & Modellierung

Empfohlene Bücher zur Vorlesung

-  J. Han and M. Kamber.
Data Mining: Concepts and Techniques.
Morgan Kaufmann, 2000.
-  David Edwards.
Introduction to Graphical Modelling.
Springer Texts in Statistics. Springer, New York, NY, 1995.
-  John C. Loehlin.
Latent Variable Models.
Lawrence Erlbaum Assoc Inc, 2004.
-  K.A. Bollen.
Structural Equations With Latent Variables.
John Wiley & Sons, 1989.

Regression & Statistik

Empfohlene Bücher zur Vorlesung

-  Alan Julian Izenman.
Modern Multivariate Statistical Techniques. Regression, Classification, and Manifold Learning.
Springer Texts in Statistics. Springer, 2008.
-  Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang.
Regression.
Springer, 2007.
-  Peter Bühlmann and Sara van de Geer.
Statistics for High-Dimensional Data.
Springer Series in Statistics. Springer, 2011.
-  J. Kreiß and G. Neuhaus.
Einführung in die Zeitreihenanalyse.
Springer, 2006.
-  H.L. Harney.
Bayesian Inference.
Springer, 2003.

Mathematische Grundlagen

Empfohlene Bücher zur Vorlesung

-  M. Drmota, B. Gittenberger, G. Karigl, and A. Panholzer.
Mathematik für Informatik.
Heldermann Verlag, 2007.
438 Seiten fester Einband 36 EUR.
-  L. Dümbgen.
Stochastik für Informatiker.
Springer, 2003.
-  G.H. Golub and C.F. Van Loan.
Matrix Computations.
Johns Hopkins University Press, 1996.
-  I.N. Bronstein and K.A. Semendjajew.
Taschenbuch der Mathematik.
Verlag Harri Deutsch, Thun und Frankfurt/Main, 24 edition, 1989.

Andere Sprachen & Systeme

Empfohlene Bücher zur Vorlesung



Ian H. Witten and Eibe Frank.

Data Mining: Practical Machine Learning Tools and Techniques.

Morgan Kaufmann, 2005.

2. Auflage.



I.T. Nabney.

NETLAB. Algorithms for Pattern Recognition.

Advances in Pattern Recognition. Springer, 2001.



Dietrich W. R. Paulus and Joachim Hornegger.

Pattern Recognition of Images and Speech in C++.

Advanced Studies in Computer Science. Vieweg, Wiesbaden, 1997.



Willi-Hans Steeb.

The Nonlinear Workbook.

World Scientific, 2005.

Paperback 29 Pound.

WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 11. April 2018

Teil I

Einführung in die Sprache 

Was kann 'R'?

Sprachumfang

Grafik

Installation

'R'-Sitzung

?plot

Softwarewerkzeuge und ihre Benutzerschnittstelle

Graphikorientiert (GUI)
Menüauswahl
Formulareintrag
Drag & Drop

Befehlsorientiert (CLI)
Fixiertes Kommandoensemble
Aufrufparameter und -syntax
Skriptunterstützung

Sprachorientiert (PLI)
Formale Anwendungssprache
Interpreterprogramm
BS/PS-Interface

Programmorientiert (API)
Unterprogrammbibliothek
Klassenbibliothek
(wie PLI) erweiterbar

Was kann 'R'?

Sprachumfang

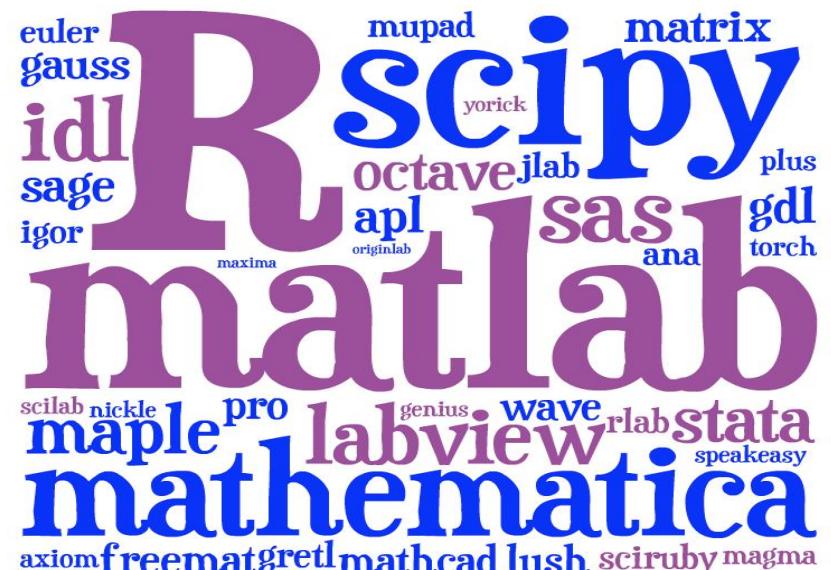
Grafik

Installation

'R'-Sitzung

?plot

Programmiersprachen für wissenschaftliches Rechnen



Inhaltlicher Vorlesungsaufbau

Die Programmiersprache 'R'

- **Sprache 'R' im Überblick**
Hilfe, Installation, Sitzung
- **Rechnen und Datentypen**
Felder, Listen, Funktionen, Operatoren
- **Text und Daten**
Zeichenketten, Eingabe/Ausgabe, Graphikausgabe
- **Statistische Modelle**
Regression, Optimierung, Verteilungen

Anwendungsgebiete von 'R'

- **Zeitreihen**
Trend/Saison, Faltung, ACF, Spektrum, AR/MA, GARCH
- **Transformationen**
Hauptachsen, MDS, Faktoren, ICA, NMF, LDA
- **Klassifikatoren**
k-NN, MLP, SVM, SCT, OLS/GLS, LDA/QDA
- **Clusteranalyse**
divisiv/agglomerativ, K-Means, Fuzzy, EM-Entmischung, spektral

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle



Das System

- ein leistungsfähiger Taschenrechner
- ein vektor/tensorbasierter Mathematikprozessor
- eine funktional-objektorientierte Programmiersprache
- ein Werkzeug zur statistischen Datenanalyse
- ein mächtiges Kommandosystem für wiss. Grafiken
- ein Interpreter für „die Sprache C ∪ Lisp“
- ein offenes und kostenloses Softwareprodukt (S-Plus nicht!)
- in den Bereichen Statistik, Datamining, Bioinf weit verbreitet
- ein Algorithmen-Wiki für die internat. Datamining-Szene



Wissenswertes über

- Dokumente des 'R'-Pakets: <file:///usr/lib/R/doc/html>
Tutorial, Language Definition, Installation, Erweiterungspakete, Datenim-/export, FAQ, ...
- Webseiten des 'R'-Projekts:
international: <http://www.r-project.org/> · national: <http://www.r-project.de/>
- Ausgewählte Handbücher und Kursmaterialien
Contributed Documentation (≥ 50 Tutorials): <http://cran.r-project.org/other-docs.html>
- Interaktive Hilfe zur Programmalaufzeit
(*Funktionsaufruf*)
 1. Funktionsbeschreibung
 2. Beispieldemonstration
 3. Funktionsdeklaration
 4. Suchfunktion

```
funcname (a,b,...)
help (funcname)
  bzw. ?funcname
example (funcname)
  funcname
help.search (topic)
  bzw. ??topic
```

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

- 'R' ist interaktiv: der Programmtext wird interpretiert
- 'R'-Programm $\hat{=}$ Folge von Ausdrücken
- der Ausdruck wird ausgewertet; der Wert wird angezeigt:
`(17+4)*(35-25)+4501 ... [1] 4711`
- nach Namensbindungen wird die Wertausgabe unterdrückt:
`xval <- (17+4)*(35-25)+4501 ... (nix)`
- der Wert wurde aber errechnet und gemerkt:
`xval + 1111 ... [1] 5822`
- Bezeichner werden nach Erstauftaft automatisch registriert:
`objects() ... [1] 'DNA.prom' 'my_url' 'xval'`
- Bezeichner sind **keine** Variablen:
`xval <- "School's out forever!" ... (kein Problem)`

Vektoren, Matrizen, Tensoren

- Die elementaren Datentypen von 'R' sind Vektoren:
`x <- c(1,4,9,16) ; x ... [1] 1 4 9 16`
- Es gibt zahlreiche Konstruktoren für Vektorobjekte:
`y <- 1:4 ; y ... [1] 1 2 3 4`
- Arithmet./logische Verknüpfungen operieren komponentenweise:
`x+y ... [1] 2 6 12 20`
- Matrizen bestehen aus Datenvektor & Strukturinformation:
`matrix (data=1:12, nrow=3, ncol=4, byrow=FALSE)`

$$\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

- Es gibt zahlreiche Werteselektionsmöglichkeiten:
`mat[3] mat[2,3] mat[2,1:3] mat[2:3,2]`

$$\begin{bmatrix} 3 & 8 \\ 2 & 5 & 8 \\ 1 & 2 & 5 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 5 & 6 \end{bmatrix}$$

Listen, Ausdrücke & Parsebäume

- Listen enthalten Komponenten von unterschiedlichem Typ:
`lis <- list (zahl=12, TRUE, 'zwei Wörter')`
- Selektion per Index oder per Name:
`lis[[2]] ... [1] TRUE oder lis$zahl ... [1] 12`
- Programmtexte können geparsst werden:
`ex <- parse (text = "a-b; (17+4)*45; x+y")`
- Es wird eine Liste von Parsebäumen erzeugt:
`ex2 <- ex[[2]] ; ex2 ... (17 + 4) * 45`
- Jeder Ausdruck liegt als Kantorovic-Ableitungsbaum vor:
`ex2[1] ex2[2] ex2[3] ex2[4]`

$$\begin{array}{c} * \\ (17 + 4) \\ 45 \\ \text{NULL} \end{array}$$
- Kantorovic-Bäume können (u.U.) ausgewertet werden:
`eval(ex2) ... [1] 945`
`eval(ex) ... Error in eval(expr, envir, enclos) : Object 'a' not found`

Datensätze laden — auch über das Internet

- Als Dateinamen werden auch URLs akzeptiert:

```
myurl <- 'http://stat.ethz.ch/Teaching/Datasets/NDK/sport.dat'
d.sport <- read.table(myurl)
```

- Datensätze mit benannten Mustern & Merkmalen:

	weit	kugel	hoch	disc	stab	speer	punkte
OBRIEN	7.57	15.66	207	48.78	500	66.90	8824
BUSEMANN	8.07	13.60	204	45.04	480	66.86	8706
DVORAK	7.60	15.82	198	46.28	470	70.16	8664
FRITZ	7.77	15.31	204	49.84	510	65.70	8644
HAMALAINEN	7.48	16.32	198	49.62	500	57.66	8613
NOOL	7.88	14.01	201	42.98	540	65.48	8543
ZMELIK	7.64	13.53	195	43.44	540	67.20	8422
GANIYEV	7.61	14.71	213	44.86	520	53.70	8318
PENALVER	7.27	16.91	207	48.92	470	57.08	8307
HUFFINS	7.49	15.57	204	48.72	470	60.62	8300
PLAZIAT	7.82	14.85	204	45.34	490	52.18	8282
MAGNUSSON	7.28	15.52	195	43.78	480	61.10	8274
SMITH	7.47	16.97	195	49.54	500	64.34	8271
MUELLER	7.25	14.69	195	45.90	510	66.10	8253
CHMARA	7.75	14.51	210	42.60	490	54.84	8249

- Auswahl von Zeilen und Spalten, z.B. d.sport[, 'kugel']

```
[1] 15.66 13.60 15.82 15.31 16.32 14.01 13.53 14.71 16.91 15.57 14.85 15.52
[13] 16.97 14.69 14.51
```

Aufrufen & Deklarieren von Funktionen

- Funktionsdeklaration mit formalen Parametern:

```
funcname <- function (arglist) {body}
```

- Parameterbezeichner *vname*
- Parameter mit Voreinstellung *vname=default*
- Restparameterliste ...

- Funktionsaufruf mit aktuellen Parametern:

```
funcname(arglist)
```

- Positionelle Übergabe *expr*
- Namentliche Übergabe *vname=expr*
- Restparameterübergabe ...

- Beispiel:

```
zeichne <- function (x, y=NULL, title='Grafik', ...) {
  if (is.null (y))
    { y <- x; x <- 1:length(x) }
  plot (y ~ x, main=title, ...)
}

zeichne (sin(1:20), cos(1:20), title='Kreis', col='red')
```

Was kann 'R'?

Sprachumfang

Grafikausgabe

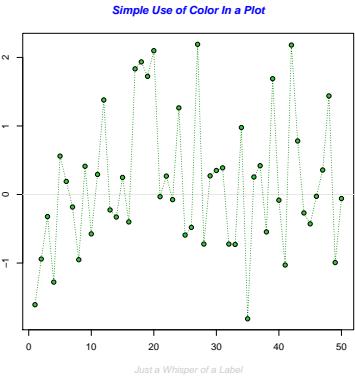
Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

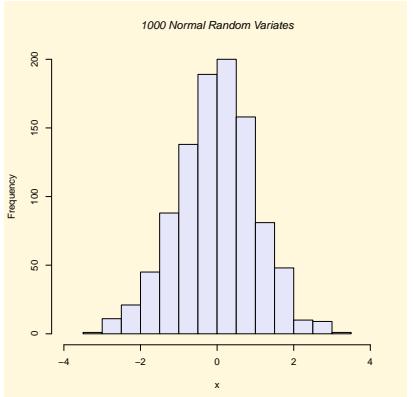
Elementare Grafikbefehle

Wertesequenzen & Wertemengen

Skalare Datensammlung mit/ohne Reihenfolgeinformation



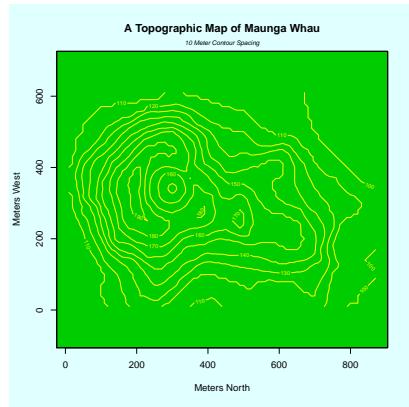
Stützwertediagramm



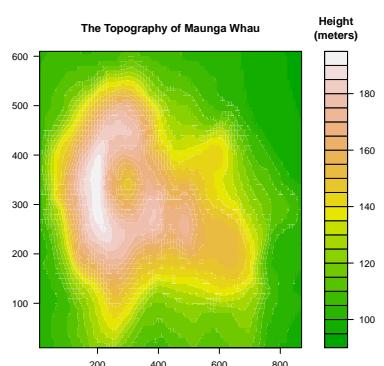
Balkendiagramm

Datenmatrizen, Einkanalbilder II

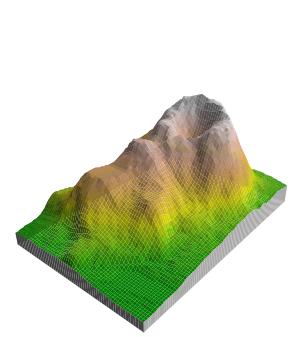
Werte durch Grenzen · Werte durch Farben



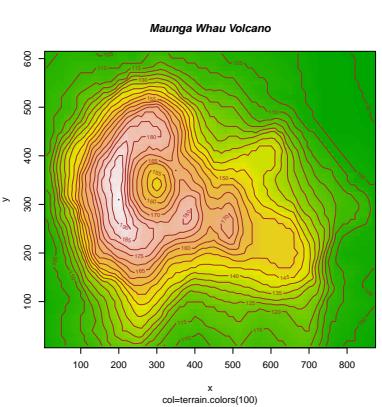
Konturen/Isolinien



Falschfarbendarstellung



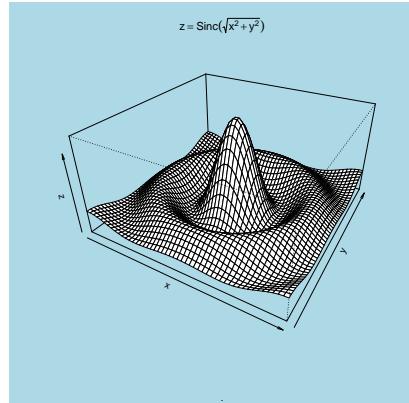
Schattierung & Falschfarben



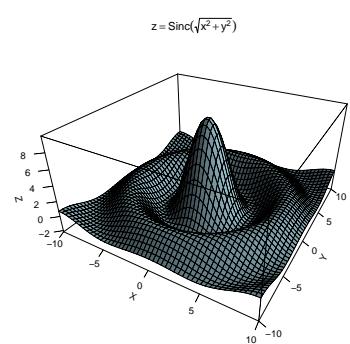
Isolinien & Falschfarben

Funktionsverläufe $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ zweier Variablen

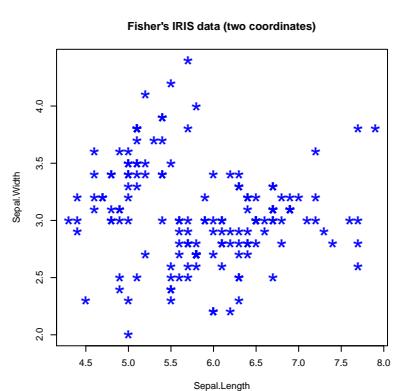
Zuordnungsvorschrift statt Datenmenge



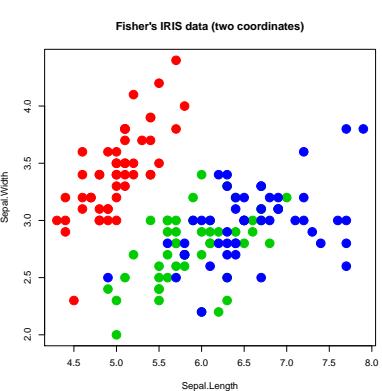
Drahtgitterlandschaft



Gitter & Schattierung



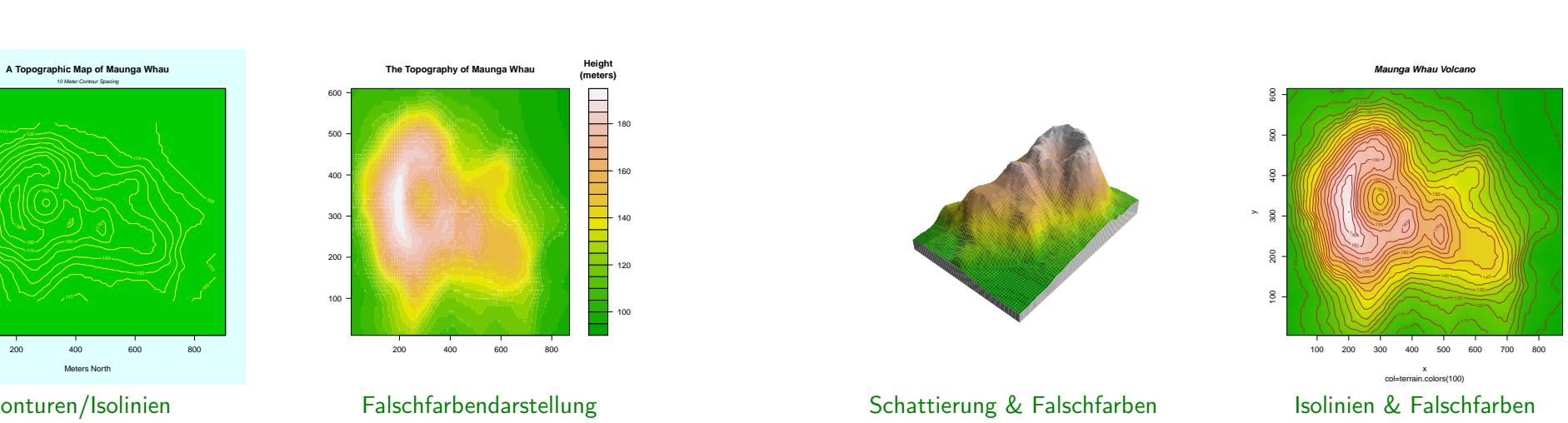
alle Datenpunkte (x_i, x_j) in der Ebene



dto., mit Klassenmarkierung

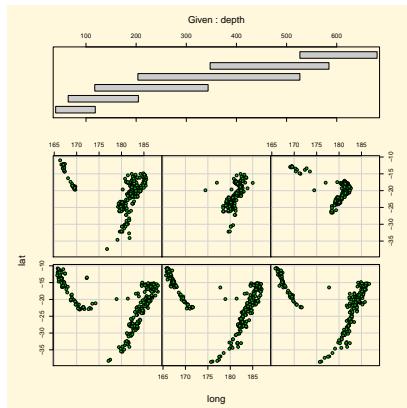
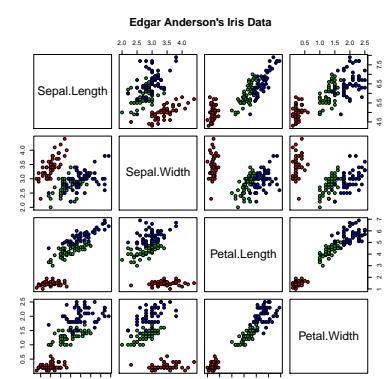
Kombinierte Skalendarstellung

Zwei Modalitäten in einem Koordinatensystem



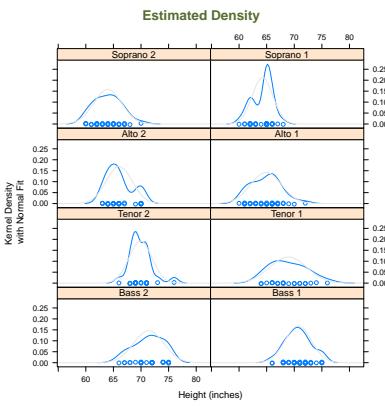
Drei- und mehrkanalige Daten

Schubladengrafik · Kombinatorischer Scatterplot

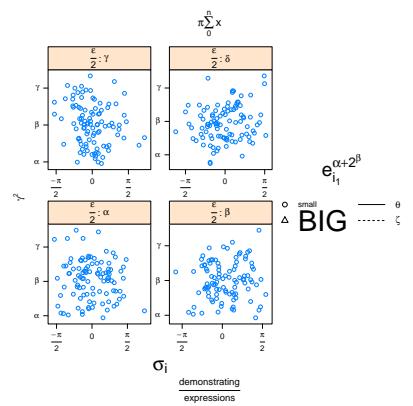
 x_3 -partitionierte (x_1, x_2) -Diagrammealle möglichen (x_i, x_j) -Diagramme

Darstellung von Verteilungsmodellen

Empirische/geschätzte Dichte · Text- und Formelsatz



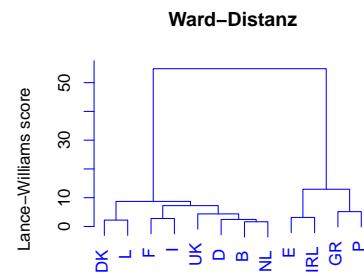
Gruppenbezogene Dichtevisualisierung



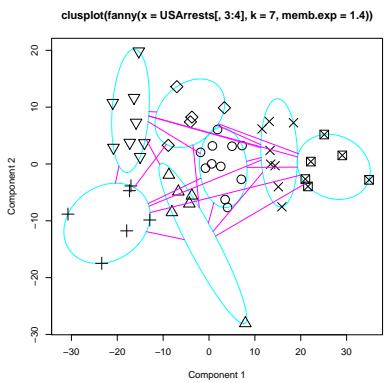
Formelsatz in LATEX-Syntax

Clusteranalyse (Gruppierung)

Hierarchische Gruppierung · Unscharfe Gruppierung



Dendrogramm (AGNES)



Partition (fuzzy K-means)

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle



R am Institut für Informatik

'R' ist verfügbar an allen Rechnern im Linux-PC-Pool des FRZ

myself@mipool> R

R version 3.3.1

```
R version 3.3.1 (2016-06-21) - "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-suse-linux-gnu (64-bit)
```

```
R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.
Tippen Sie 'license()' or 'licence()' für Details dazu.
```

```
R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()', um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.
```

```
Tippen Sie 'demo()', für einige Demos, 'help()', für on-line Hilfe, oder 'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.
```



R zu Hause unter Linux installieren

1. Rufen Sie die Webseite <http://cran.r-project.org/>
CRAN = Comprehensive 'R' Archive Network
2. Nutzen Sie die ONE-CLICK-Installation
`R binaries ~> linux ~> suse ~> SuSE 12.1 R-patched-devel-2.14.1`
3. ... oder laden Sie sich eine Distribution herunter und installieren Sie das Paket als 'root':
`rpm -i /home/schukat/hereis/R-base-2.4.0-1.i586.rpm`

```
error: Failed dependencies:
xorg-x11-fonts-100dpi is needed by R-base-2.4.0-1
blas is needed by R-base-2.4.0-1
libgfortran.so.0 is needed by R-base-2.4.0-1
```

4. Installieren Sie die fehlenden Pakete nach ([YAST2](#)), z.B.
`blas · xorg-x11-fonts-100dpi · fortran o.ä.`



R zu Hause unter Windows installieren

Warum sollte das eigentlich **nicht** möglich sein? · Wer hat da eben gelacht?

(Algorithmus)

- 1 Laden Sie sich eine Binärversion für Windows XP herunter.
- 2 Überzeugen Sie den Installationsassistenten Ihres Vertrauens davon, daß es sich beim 'R'-Paket um ein Egosshooterprogramm handelt.
- 3 Führen Sie einen Systemneustart durch.
- 4 Entfernen Sie die verkohlten Kunststoffreste von der Auslegeware.

(zumDingA)

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

Elementare Grafikbefehle

Beginn und Ende einer 'R'-Sitzung

Konfiguration einer individuellen Umgebung

- Starten einer Sitzung:
`.../working-directory> R`
- Beenden einer Sitzung:
`quit()` oder `q()` oder `q('yes')` oder `q('no')`
- Der individuelle Zuschnitt Ihrer 'R'-Umgebung:
`.../working-directory/.Rprofile`
- Prolog zu Sitzungsbeginn:
... die Funktion `.First()` wird ausgeführt
- Epilog bei Sitzungsende:
... die Funktion `.Last()` wird ausgeführt
- BEISPIEL:
Die Startdatei `.Rprofile` enthält die Funktionsdefinitionen:
`.First <- function() { source ('./lib/mydefs.R') ; cat (paste (date(), 'Hola')) }`
`.Last <- function() { graphics.off() ; cat (paste (date(), 'Adios')) }`

Kommandos zur Objektverwaltung

Listen · Löschen · Ein/Ausgabe von Daten und Kommandos

- Auflisten aller definierten Objekte:
`objects()` oder `ls()`
- Löschen definierter Objekte:
`remove (x,y,fun)` oder `rm (list=ls())` (`!?`)
- 'R'-Kommandos von Datei lesen
`source (file)`
(Eingabeumlenkung)
- 'R'-Ausgaben/Meldungen in Datei schreiben
`sink(file=NULL, append=FALSE)`
(Ausgabeumlenkung)
- 'R'-Objekte in eine Datei speichern
`save(..., list=character(0), file=)`
- 'R'-Objekte aus einer Datei laden
`load(file)` oder `loadURL(url)`
(char vector)
- Alle Sitzungsobjekte werden nach `.RData` geschrieben

Stapelverarbeitung für -Programme

Aufruf von und Parameterübergabe an 'R'-Skriptdateien

- Abarbeitung einer Datei mit 'R'-Programmcode:

`Rscript progfile.R`

(entspricht Ausführung von `source ("progfile.R")` im 'R'-Interpreter)

- Aufruf eines 'R'-Skripts mit Parameterübergabe:

`Rscript progfile.R string1 string2 string3`

Nach der Zuweisung

```
arg <- commandArgs (trailingOnly=TRUE)
```

finden sich in den Komponenten `arg[1]`, `arg[2]` usw. des `character`-Vektors `arg` die Aufrufparameter als Zeichenketten.

- Etwaige Grafikausgaben finden wir im aktuellen Arbeitskatalog unter dem Namen `Rplots.pdf`.

(sofern „PDF“ voreingestellt und kein Dateiname angegeben)

Was kann 'R'?

Sprachumfang

Grafikausgabe

Installation des 'R'-Pakets

Ablauf einer 'R'-Sitzung

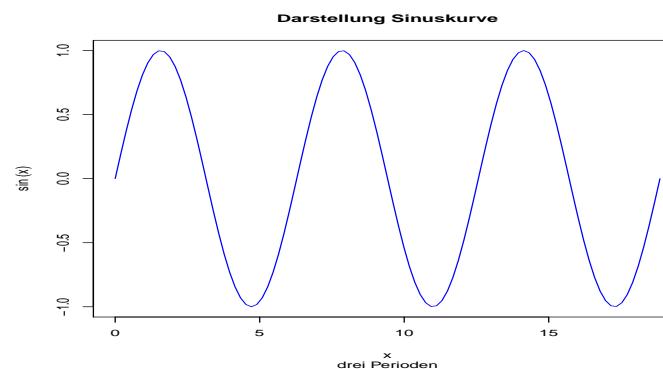
Elementare Grafikbefehle

Zeichnen von Funktionsverläufen

- Generische Funktion delegiert an zuständige Methode:
`plot(func,...) ~> plot.function(func,...)`
- Methode zum Kurvenzeichnen:
`curve(expr, from, to, n=101, add=FALSE, type='l',
ylab=NULL, log=NULL, xlim=NULL, ...)`
- Der Funktionsname ist gegeben, z.B.:
`plot(sin)`
- Eine Funktionsdefinition ist gegeben, z.B.:
`plot(function(z) {2*z+5})`
- Das Intervall für die Argumentwerte ist spezifiziert, z.B.:
`plot(sin, -2, +4)`

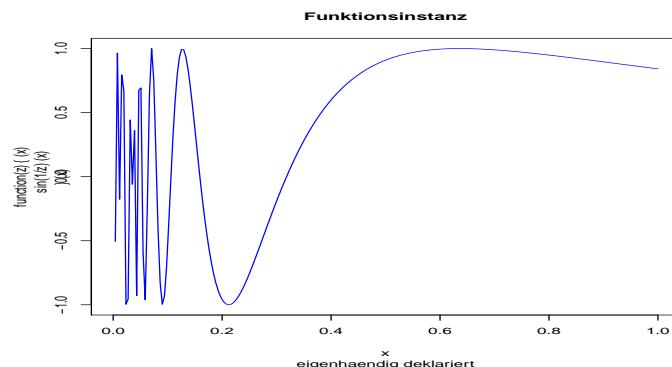
Beispiel: eine Sinuskurve

```
plot ( sin, from=0, to=6*pi, col="blue",
       main="Darstellung Sinuskurve",
       sub="drei Perioden" )
```



Beispiel: eine Funktionsdeklaration

```
plot ( function(z) {sin(1/z)}, col='blue', n=256,
       main='Funktionsinstanz',
       sub='eigenhändig deklariert' )
```

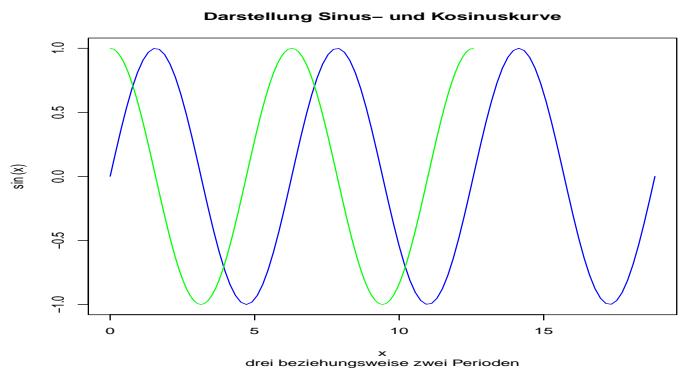


Bildübergreifende Grafikbefehle

- Überlagern mehrerer Verläufe in einem Achsenkreuz
`plot (x,y, add=TRUE)`
- Umleiten der Grafikausgabe von Schirm auf Datei
`pdf (file='Rplot.pdf', ???)`
 und entsprechend für `postscript pictex png jpeg GTK GNOME xfig bitmap`
 zurück mit `X11 (display, ???)`
- Keine neue Zeichnung ohne Bestätigung
`par (ask=TRUE)` (Stapelbetrieb)
- Aufteilung der Leinwand in mehrere Grafikfelder
`par (mfrow=c(n,m))` oder
`par (mfcol=c(n,m))`

Beispiel: zwei Kurven in einem Bild

```
plot ( sin, from=0, to=6*pi, col="blue",
       main="Darstellung Sinus- und Kosinuskurve",
       sub="drei beziehungsweise zwei Perioden")
plot ( cos, from=0, to=4*pi, add=TRUE, col="green")
```



Wertesequenzen und Punktesequenzen

- Wertesequenzen zeichnen: `plot(x)`, z.B.:

```
plot (c(2,3,5,7,11,13,17))
plot (1:20)
plot (seq(0,8,0.1))
plot (sin (seq(0,8,0.1)))
```

- Punktesequenzen zeichnen: `plot(x,y)`

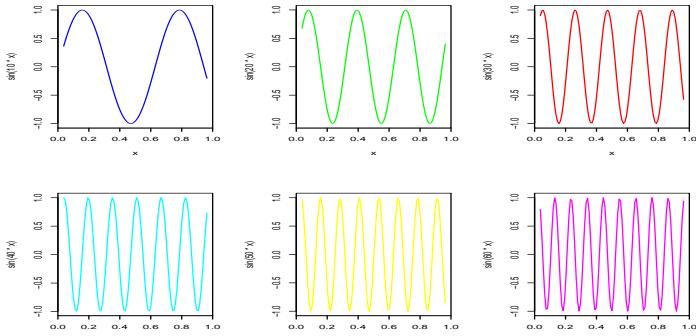
z.B. `x <- seq(0,2*pi,0.1)` nebst
`plot(x, sin(x))` oder
`plot(sin(x), x)` oder
`plot(cos(x), sin(x))`

- Kurvendarstellung durch `type='?'` gesteuert:

```
points lines both
overplotted contour-only
high-density steps Steps ...
```

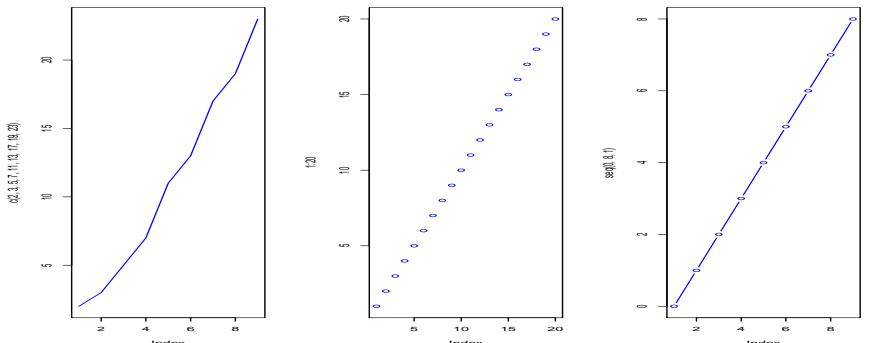
Beispiel: sechs Bilder auf einem Blatt

```
par (mfrow = c(2,3))
curve (sin(10*x), col="blue"); curve (sin(20*x), col="green")
curve (sin(30*x), col="red"); curve (sin(40*x), col="cyan")
curve (sin(50*x), col="yellow"); curve (sin(60*x), col="magenta")
```



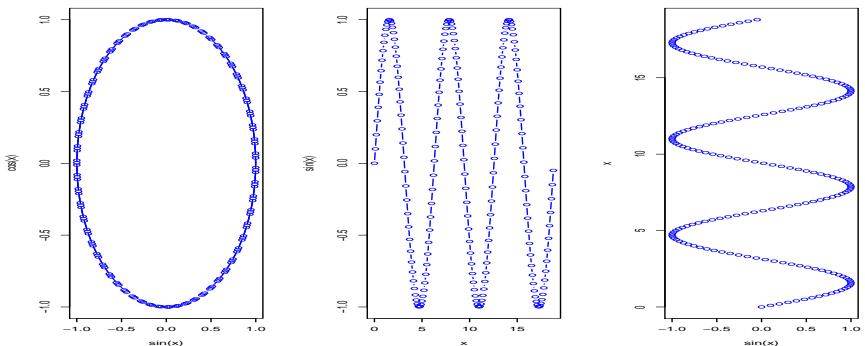
Beispiel: eindimensionale Wertefolgen

```
par (mfrow = c(1,3))
plot (c(2,3,5,7,11,13,17,19,23), type='l', col='blue')
plot (1:20,
      type='p', col='blue')
plot (seq(0,8,1),
      type='b', col='blue')
```



Beispiel: zweidimensionale Wertefolgen

```
par (mfrow = c(1,3))
x <- seq (0, 6*pi, 0.1)
plot (sin(x), cos(x), type='b', col='blue')
plot (x, sin(x), type='b', col='blue')
plot (sin(x), x, type='b', col='blue')
```

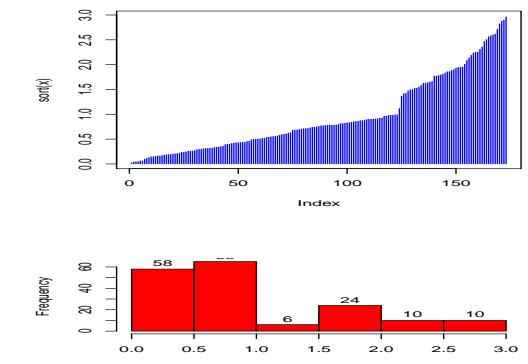


Zusammenfassung (1)

1. 'R' ist eine **funktionale** Programmiersprache mit rudimentärer Objektorientierung (**polymorphe** Aufrufe).
2. 'R' arbeitet **interaktiv** (Interpreter statt Compiler), unterstützt aber Stapelverarbeitung (**Skripten**).
3. 'R' gilt als spektakülär für wissenschaftliches Rechnen, **Statistik**, **Datenmodellierung** und -**visualisierung**; für Bild- und Textverarbeitung gibt es performantere Wettbewerber.
4. Der elementare Datentyp von 'R' ist der **Vektor**.
5. In 'R' sind auch **Funktionen** und **Ausdrücke** Sprachobjekte, können also zur Laufzeit manipuliert werden.
6. 'R' unterstützt die E/A beliebiger 'R'-Objekte durch **automatische Serialisierung**.
7. **Grafiken** werden kumulativ erzeugt; zuerst das **Koordinatensystem**, dann sukzessive weitere **Zeichnungskomponenten**.
8. Komplexere Grafiken werden durch **Leinwandaufteilung** und sequentielle **Komposition** einfacher Grafiken erzeugt.

Leinwandaufteilung mit variablen Feldmaßen

```
layout (matrix (c(1,1,2,3), ncol=2), width=c(2,3), height=c(3,2))
x <- c (runif (123), rnorm (50, mean=2, sd=0.5))
boxplot (x, col="green")
plot (sort(x), col="blue", type="h")
hist (x, col="red", lab=TRUE, main="")
layout (1)
# nicht vergessen!!
```



WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Teil II

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 30. Mai 2018

Rechnen in 

'R'-Syntax ?vector ?matrix ?list ?data.frame ?factor ?Control function() class()

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

'R'-Syntax ?vector ?matrix ?list ?data.frame ?factor ?Control function() class()

Syntax und Semantik

Übersetzte vs. interpretierte Programmiersprachen

Syntax

Ist der Quellcode ein zulässiger Programmtext oder nicht?

Semantik

Was wird berechnet?
Was wird bewirkt?

Syntaxfehler

{ keine Übersetzung!
 kein Start! }

Laufzeitfehler

Wert oder Zustand undefiniert!

Übersetzung

Compiler transformiert Quellcode in Zielcode.

Vor- & Nachteile

⊕ Effizienz
⊖ Modifikation

Interpretation

Interpreter führt Quellcode schrittweise aus.

Vor- & Nachteile

⊕ Interaktion
⊖ Effizienz



ist interpretativ und funktional

IM PRINZIP EINFACH ...

Funktionale Programme

bestehen aus einer Folge von Ausdrücken.

Ausdrücke

sind geschachtelte Funktionaufrufe.

Auswertung

eines Ausdrucks immer von innen nach außen.

Anweisungen

gibt es nicht.

IM DETAIL KOMPLEXER ...

- Zeilenorientierung
- Variable & Namensräume
- Explizite Typanpassung
- Operatorschreibweise
 - Arithmetik & Logik
 - Indexnotation (Reihung)
 - Selektion (Verbund)
 - Zuweisung
 - Datenmodelle

- teilweise Klammerzwang
- explizite Auswerteregelung
- Verzögerung & Versprechen
- „Kontrollstrukturen“

Die Operatoren von



Syntaktischer Zucker & seine Bindungsfähigkeit

I	\$	Komponentenauswahl	list\$item		
II	[[Elementzugriff	x[i] A[5,3] df\$p[5]		
III	^	Exponentiation	x^3		
IV	-	Minusvorzeichen	-08.15 -5^2		
V	:	Indexfolgen	1:8 5:-3 -5:3		
VI	%op%	benutzerdefiniert	x %*% y 10%% 2		
VII	*	/	Punktrechnung	8*4 21/7 883+0:5	
VIII	+	-	Strichrechnung	17+4 x[n-1] a*x+b	
IX	<	<=	Vergleichs-	1+1 != 3	
	>	>=	operatoren	(x<y) == (y>x)	
X	!		Negation	(!3==1) == TRUE	
XI	&	&&	Konjunktion	p & (q r) !a b	
			Disjunktion	is.vector(x) && plot(x)	
??	~		Modellformel	z ~ (x1+x2):(z1+z2)	
XII	<-	->	=	Zuweisung	13->x names(x)<- "Kevin"

Die Syntax von



Aus großer Flughöhe bei hoher Geschwindigkeit beobachtet

Programm = Folge von Ausdrücken

```

program ::= expr*
expr ::= objID | literal | funcall | fobj | (expr) | block | control
block ::= {[expr exsep]* expr]}
control ::= if1Ctl | if2Ctl | forCtl | repeatCtl | whileCtl
exsep ::= ; | \n | exsep+

```

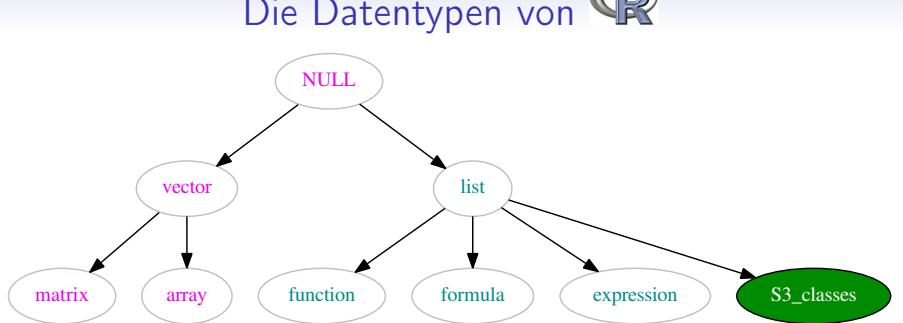
Funktionsaufruf (Standard- oder Operatorform)

```

funcall ::= fname(arglist) | opcall
arglist ::= {[arg ,]* arg}
arg ::= expr | formpar=expr | ...
opcall ::= unop expr | expr binop expr | expr[i list] | exp[[idx]]

```

Die Datentypen von



Elementare und ...

logical()
integer()
numeric()
complex()
character()

komplexe Datentypen (Verbundobjekte)

Liste = Elemente + Attribute

Rudimentäre Objektorientierung:

- Hierarchisches Klassenattribut (Spezialisierungspfad)
- Polymorpher Methodenaufruf (Argument #1)

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Atomare Datentypen

Es gibt **keine** skalaren Typen in 'R' — nur Vektoren der Länge 1

- **Datentyp `NULL`** „*undefined*“
exkl.: `NULL`
- **Datentyp `logical`** Wahrheitswerte
z.B.: `TRUE`, `FALSE` oder `NA`
- **Datentyp `numeric`** ganze und Gleitkommazahlen
z.B.: `17`, `3.14`, `-1.2e6`, `pi`, `NaN`, `Inf`, `NA`
- **Datentyp `complex`** komplexe Zahlen
z.B.: `2.13+1i`, `0-47.11i`, `2e-7i` oder s.o.
- **Datentyp `integer`** ganze Zahlen
z.B.: `1:12`, aber nicht `17` usw.
- **Datentyp `character`** Buchstaben und Zeichenfolgen
z.B.: `"Hello World!"`, `'a'`

Typabfrage und Typkonversion · Literale

• Typidentifikation

```
mode (3.141593)
[1] 'numeric'
```

• Typverifikation

```
is.character (47.11)
[1] FALSE
is.numeric (47.11)
[1] TRUE
is.complex (47.11)
[1] FALSE
```

• Typkonversion

```
sqrt (as.complex (-1))
[1] 0+1i
sqrt (as.numeric (-1))
[1] NaN
```

• Nullreferenz

<code>is.null (NULL)</code>	<code>NULL</code>
<code>is.null (list())</code>	<code>T</code>
<code>is.null (integer(0))</code>	<code>F</code>

• Fehlanzeige

<code>is.na (NA)</code>	<code>NA</code>
<code>is.na (883)</code>	<code>T</code>

• Unendlich

<code>is.finite (pi)</code>	<code>Inf</code>
<code>is.infinite(pi/0)</code>	<code>T</code>
<code>is.infinite(1/0+1/0)</code>	<code>T</code>
<code>is.infinite(1/0-1/0)</code>	<code>F</code>

• Undefiniert

<code>is.nan (0/0)</code>	<code>NaN</code>
<code>is.nan (1/0-1/0)</code>	<code>T</code>

• Konstruktor

```
vector (mode='numeric', length=12)
```

• Konkatenation

```
c (2,3,5,7,11,13,17)
c (12, c(4,5,6), 7, v4)
```

• Arithmetische Progressionen

```
1:8, 5:2
seq (1,17, by=2)
seq (1,17, length.out=50)
seq (along.with=c(2,3,5,7,11,13)) besser als 1:length(x)
```

• Wiederholung von Elementen und Folgen

<code>rep (x, times=5)</code>	wie <code>rep(x,5)</code> oder <code>c(x,x,x,x,x)</code>
<code>rep (x, length.out=17)</code>	für <code>length(y) > 1</code>
<code>rep (x, each=5)</code>	
<code>rep (x, times=y)</code>	

Indizierter Zugriff auf Vektoren

- **Indexmenge** = $\{1, 2, \dots, \ell\}$

`length(x)`
`length(x) <- lnew` (kürzen oder mit `NA`s auffüllen)

- **Einzelelemente**

`x[5]`

- **Elementfolgen selektieren**

`x[c(3,4,7)]`

`x[3:5]`

- **Elemente unterdrücken**

`x[c(-3,-4,-7)]`

`x[-c(3,4,7)]`

Syntaxfehler: `x[c(+2,-3)]`

- **Logische Indizierung**

`(1:5) [c(FALSE,TRUE,TRUE,FALSE,TRUE)]` $\rightsquigarrow 2 \ 3 \ 5$

`(1:50) [c(FALSE,TRUE)]` $\rightsquigarrow 2 \ 4 \ 6 \ 8 \ 10 \ 12 \ 14 \dots$

`gehalt [gehalt > 78000]`

Funktionen und Operatoren

- **Funktionen von \mathbb{R}^n nach \mathbb{R}**

Summe/Produkt
Extremwerte
Statistik

`sum()`, `prod()`
`max()`, `min()`
`mean()`, `median()`, `var()`, `sd()`

- **Funktionen von \mathbb{R}^n nach \mathbb{R}^n**

Logarithmen
Rundung `round()`, `signif()`, `trunc()`, `floor()`, `ceiling()`
Trigonometrie
Sonstige
Kumulative Berechnungen

`log()`, `log10()`, `log2()`
`exp()`, `sqrt()`, `abs()`
`cum{sum,prod,max,min}()`

- **Operatoren (zweistellige Funktionen in Infixschreibweise)**

Strich- und Punktrechnung
Potenzbildung
Ganz Zahldivision/Rest

`x+y`, `x-y`, `x*y`, `x/y`
`x^y` oder `x**y`
`x %/% y` bzw. `x %% y`
`+' (1:2, 4:5) ~> 5 7`

Beispiel:

Vektorkomponenten mit Namen

- **Konstruktion benannter Vektoren**

`x <- c(karl=6, heinz=28, mandy=17,)`

- **Namenlisten sind Vektorattribute**

`namelist <- names(x)` (Typ `character`)
`namelist <- attr(x, 'names')` (`dto.`)

- **Namen können geändert oder gelöscht werden**

`names(x)[2] <- 'osama'` (einzeln)
`names(x) <- NULL` (alle)

- **Komponenten lassen sich durch Namen indizieren**

`x['mandy'] == x[3]` $\rightsquigarrow \text{TRUE}$
`x[rep('karl',3)]` $\rightsquigarrow 6 \ 6 \ 6$

Komplexwertige Vektoren

Konstruktor `complex()` und Projektionen `Re()`, `Im()`, `Mod()`, `Arg()` und `Conj()`

- **Konstruktion komplexer Vektoren**

`complex (3)` $0+0i \ 0+0i \ 0+0i$
`c (3i+2, 1-5i)` $2+3i \ 1-5i$
`sqrt (-1:+1 + 0i)` $0+1i \ 0+0i \ 1+0i$
`is.complex (-08.15)` FALSE

- **Projektion komplexer Zahlen**

<code>Re (3.0-4.0i)</code>	3
<code>Im (3.0-4.0i)</code>	-4
<code>Mod (3.0-4.0i)</code>	5
<code>Arg (3.0-4.0i)</code>	-0.9272952
<code>all.equal (sin (Arg (3.0-4.0i)), -4/5)</code>	TRUE
<code>Conj (3.0-4.0i)</code>	3+4i

Wahrheitswerte

Vektoren vom Typ logical

- **Dreiwertige Logik in 'R'**

`TRUE, FALSE` und `NA`

Literale

- **Vergleichsoperatoren**

`==, !=, <=, >=, <, >`

arithmetisch oder lexikographisch

- **Logische Verknüpfungen**

`!b`
`a&b, a|b, xor(a,b)`
`all(b), any(b)`

einstellig
zweistellig; *nicht* strikt!
 \forall/\exists -Quantor

- **Bedingte Ausdrücke**

`if (b) x else y`
`ifelse (b, x, y)`

skalare Wertverzweigung
vektoruelle Wertverzweigung

- **Selektion von Unterfeldern**

`(1:8)[c(TRUE, FALSE)] == c(1, 3, 5, 7)` TRUE
`lebensabendspanne <- mean(alter[alter>65]) - 65`

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Vektorisierte Suchoperationen

Trefferpositionen und -meldungen für Vergleichsanfragen

- **Extremalposition (min/max)**

`which.max(c(4, 7, 1, 1))`
`which.min((-69:+96)^2)`

2

70

- **Logische Trefferpositionen**

`which(LETTERS=='H')`
`which(11:20 > 17)`

8

8 9 10

- **Test auf Wertegleichheit — Trefferpositionen**

`match(x=4, table=abs(-5:+5))`
`match(x=c(8, 8, 3, 0), table=12:1)`

2

5 5 10 NA

- **Test auf Wertegleichheit — Treffermeldungen**

`c(8, 8, 3, 0) %in% 12:1` TRUE TRUE TRUE FALSE
`c('S', 'RTFM') %in% LETTERS[1:20]` TRUE FALSE

Matrizen

Klasse `matrix` — Vektoren mit Dimensionsattribut

- **Konstruktoren**

`A <- diag(5)` Einheitsmatrix in $\mathbb{R}^{5 \times 5}$
`A <- diag(c(4, 7, 1, 1))` Diagonalmatrix in $\mathbb{R}^{4 \times 4}$
`matrix(data=NA, nrow=1, ncol=1, byrow=FALSE)`
`A <- matrix(1:12, ncol=3)` Matrix in $\mathbb{R}^{4 \times 3}$
`A <- matrix(1:3, 4, 3, byrow=TRUE)` Matrix in $\mathbb{R}^{4 \times 3}$

- **Dimensionsattribut**

`dim(A)` [1] 4 3
`nrow(A)` [1] 4
`ncol(A)` [1] 3
`length(A)` [1] 12
`dim(A) <- c(3, 4)` Todesstrafe!

Matrizen

Typ · Klasse · Transposition · Konversion

- **Typ und Klasse**

```
mode (A)
class (A)
is.matrix (A)
is.matrix (1:12)
```

- **Transponieren einer Matrix**

```
dim (A)
dim (t(A))
```

- **Explizite und implizite Typkonversion**

```
dim (as.matrix (1:12))
dim (t (1:12))
as.vector (diag (4))      spaltenweise angeordnet
all (c (matrix (1:12,4,3)) == 1:12) [1] TRUE
```

```
[1] 'numeric'
[1] 'matrix'
[1] TRUE
[1] FALSE
```

```
[1] 4 3
[1] 3 4
```

```
[1] 12 1
[1] 1 12
[1] TRUE
```

Matrizen

Selektion für Fortgeschrittene

- **Selektion einer beliebigen Matrixprojektion**

```
is.matrix (A [idx,jdx])
(A[idx,jdx]) [n, m] = A[idx[n],jdx[m]]
```

passende Indizes; nicht notwendig aufsteigend; evtl. Wdh.

- **Selektion mit Wahrheitswertmatrix**

```
is.matrix (A)
is.matrix (B)
is.logical (B)
all (dim(A) == dim(B))
Vektor aller TRUE-markierten Matrixelemente:
is.vector (A[B])
```

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

Matrizen

Selektion von Elementen, Zeilen, Spalten, Blöcken

- **Selektion eines Matrixelements**

```
A [i,j]
A [k]
```

Komponente A_{ij} (Zeile/Spalte)
k-tes Vektorelement nach Konversion

- **Selektion eines Matrixblocks**

```
A [i1:i2,j1:j2]
A [i1:i2,]
A [,j1:j2]
```

Block $[A_{ij}]_{i_1 \leq i \leq i_2}^{j_1 \leq j \leq j_2}$
alle Spalten
alle Zeilen

- **Automatische Dimensionsreduktion**

```
A [i,]
A [,j]
A [i,,drop=FALSE]
A [,j,drop=FALSE]
A [i,j,drop=FALSE]
```

(!) Vektor $[A_{ij}]_{j=1..nc}$
(!) Vektor $[A_{ij}]_{i=1..nr}$
einzelige Matrix
einspaltige Matrix
einelementige Matrix

Matrizen

Zeilen · Spalten · Diagonale

- **Rechnen mit Zeilen- und Spaltenindizes**

```
all (row(A)[i,] == i)
all (col(A)[,j] == j)
all (col(A) == t(row(t(A))))
A <- matrix(1:9,3); A[row(A)<col(A)]
```

TRUE für jedes i
TRUE für jedes j
TRUE
[1] 4 7 8

- **Matrixdiagonale**

```
is.vector (diag (A))
all (x == diag(diag(x)))
```

TRUE
TRUE für Vektoren

- **Zeilen untereinander stellen**

```
rbind (...)
rbind (A,B)
rbind (x,y)
rbind (A,y)
```

Vektoren und/oder Matrizen
Matrizen gleicher Spaltenzahl
kürzerer Vektor wird wiederholt
Matrix bestimmt Spaltenzahl

- **Spalten nebeneinander stellen**

```
cbind (...)
```

(analog)

Matrizen

Multiplikation von Matrizen

- Komponentenweise Matrixoperationen**

`A+B, A*B, ...`

(siehe `vector`)

- Matrixmultiplikation**

`ncol(A) == nrow(B)`

konforme Matrixdimensionen

`C <- A %*% B`

liefert $C = A \cdot B$

`nrow(C)==nrow(A)`

`TRUE`

`ncol(C)==ncol(B)`

`TRUE`

- Inversenbildung**

`X <- solve (A)`

A quadratisch, invertierbar

`all (A %*% X == diag(ncol(A)))`

i.a. nicht `TRUE`

- Lösung linearer Gleichungssysteme**

`X <- solve (A,B)`

löst das LGS $A \cdot X = B$

`all (A %*% X == B)`

i.a. nicht `TRUE`

Matrizen

Multiplikation von Vektoren und Matrizen

- Operator `%*%` berechnet immer eine Matrix**

`class (1 %*% 1)`

`'matrix'`

- Lineare Vektorabbildung**

`A %*% y`

$A \cdot y$ einspaltig

`x %*% B`

$x^T \cdot B$ einzeilig

- Inneres Vektorprodukt**

`x %*% y`

$x^T y$ einelementig (!)

- Äußeres Vektorprodukt**

`x %o% y`

xy^T dyadische Produktmatrix

`outer (X, Y, FUN='*', ...)`

Defaultfall = dto.

`x %*% t(y)`

weil `t(y)` eine Matrix ist

Lineare Algebra

QR-, Eigen- und Singulärzerlegung mit den Funktionen `qr()`, `eigen()` und `svd()`

- QR-Zerlegung**

$X=Q \cdot R$, $Q^T Q=E$, R ist OD-Matrix

`B <- matrix (1:12, nrow=3)`

`o <- qr (B)`

`all (qr.Q(o) %*% qr.R(o) == B)` `TRUE`

- (Symmetrische) Eigenwertaufgabe**

$S=U \Lambda U^T$, $U^T U=E$, $\Lambda=\text{diag}(\lambda)$

`o <- eigen (S <- B %*% t(B))`

`all (o$vec %*% diag(o$val) %*% t(o$vec) == S)` `TRUE`

(auch Rechts- und Linkseigenvektoren nichtsymmetrischer Matrizen)

- Singulärwertaufgabe**

$X=VDU^T$, $V^T V=E$, $U^T U=E$, $D=\text{diag}(s)$

`o <- svd (B)`

`all (o$v %*% diag(o$d) %*% t(o$u) == B)` `TRUE`

- Determinante (quadratische Matrix)**

`det (diag(1:5)) == prod (1:5)` `TRUE`

`determinant (diag(1:64), log=TRUE) sign=1 modulus=205.1`

`det(A) ≡ prod(eigen(A)$val) ≡ prod(diag(qr.R(qr(A))))`

Distanzmatrizen

Klasse `dist` — symmetrische Matrix mit Nulldiagonale

- Konstruktor**

`dist (x, method='euclidean', diag=F, upper=F)`

$D[i,j] = d_{\text{method}}(x[i], x[j])$

`'euclidean'`, $\|\cdot\|_2$, `'maximum'`, $\|\cdot\|_\infty$, `'manhattan'`, $\|\cdot\|_1$, `'canberra'`

- Klasse und Typ**

`all (dist(diag(5)) == sqrt (2))` [1] `TRUE`

`class (dist(diag(5)))` [1] `'dist'`

`mode (dist(diag(5)))` [1] `'numeric'`

- Konversion zwischen Matrix und Distanz**

`as.matrix (x)` redundante Quadratmatrix

`as.vector (x)` das untere Dreieck seriell

`as.dist (m, diag=F, upper=F)` unteres Dreieck

Felder beliebiger Dimension

Klasse array (data=NA, dim=length(data), dimnames=NULL)

- Konstruktor

```
a1 <- array (1:24, c(24))
a2 <- array (1:24, c(6,4))
a3 <- array (1:24, c(2,3,4))
```

1D-Feld
2D-Feld
3D-Feld

- Typprüfung

```
is.array (a3) etc.
is.vector (a1)
is.matrix (a2)
is.array (diag (5))
```

TRUE
FALSE
TRUE
TRUE

- Verallgemeinerte Transposition

```
all (aperm (a2, c(2,1)) == t (a2))
a3p <- aperm (a3, c(2,3,1))
a3[i,j,k] == a3p[j,k,i]
dim (UCBAdmissions) (Datenbeispiel)
```

TRUE
TRUE
2 2 6

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Listen

Klasse list — enthält benannte Komponenten unterschiedlichen Typs

- Konstruktoren

```
L <- list ()
L <- list (a=2,b=3,c=5)
```

die leere Liste

Liste von numeric-Vektoren

- Gemischt und geschachtelt

```
L <- list (83.5, TRUE, c('hello','world'), diag(4))
M <- list (IQ=83.5, above=L, lily=list(17,TRUE))
```

- Typprüfung

```
is.list (list (cottbus=0:3))
is.list (0:3)
```

ergibt TRUE

ergibt FALSE

- Typwandlung

```
as.list (c(2,3,5,7,11))
as.list (diag(5))
as.list (plot.vector)
```

Vektoren elementweise

Matrizen elementweise

Funktionen: , Argliste, Rumpf

Listen

Selektion von Komponenten mit '[[.....]]'

- Zugriff mit Index

```
L <- list (IQ=83.5, debil=T, lily=list(z=17,F))
L[[1]]
```

[1] 83.5

- Zugriff mit Namen

```
L[['IQ']]
L$IQ
```

[1] 83.5

dto., eleganter

- Zugriff wiederholt

```
L$lily$z
L$lily[[2]]
L[['lily']][[2]]
L[[3]][[2]]
```

[1] 17

[1] FALSE

[1] FALSE

[1] FALSE

- Zugriff wiederholt mit Indexvektor

```
L[[c(3,2)]]
L[[c('lily','z')]]
```

[1] FALSE

[1] 17

Listen

Selektion von Teillisten mit '[.....]',

- Zugriff mit Indexfolge**

```
L <- list (IQ=83.5, debil=T, lily=list(z=17,F))
L[1:2]                                wie list (IQ=83.5, debil=T)
L[c(2,1)]                                wie list (debil=T, IQ=83.5)
```

- Vorsicht: Einerlisten**

```
L[1]                                wie list (IQ=83.5)
L[3]                                wie list (lily=list(z=17,F))
```

- Löschen von Listenelementen**

```
M <- list (a=1,b=2,c=3)
M$c <- NULL                          ergibt list (a=1,b=2)
M$b <- NULL                          ergibt list (a=1,c=3)
(Index bleibt fortlaufend ununterbrochen!)
```

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Dataframes

Beispiel: '<http://stat.ethz.ch/Teaching/Datasets/NDK/sport.dat>'

- Leseroutine akzeptiert Dateinamen, Eingabeströme und URLs

```
d.sport <- read.table ('sport.dat')
```

- Datensätze mit benannten Mustern & Merkmalen:

	weit	kugel	hoch	disc	stab	speer	punkte
O'BRIEN	7.57	15.66	207	48.78	500	66.90	8824
BUSEMANN	8.07	13.60	204	45.04	480	66.86	8706
DVORAK	7.60	15.82	198	46.28	470	70.16	8664
FRITZ	7.77	15.31	204	49.84	510	65.70	8644
HAMALAINEN	7.48	16.32	198	49.62	500	57.66	8613
NOOL	7.88	14.01	201	42.98	540	65.48	8543
ZMELIK	7.64	13.53	195	43.44	540	67.20	8422
GANIYEV	7.61	14.71	213	44.86	520	53.70	8318
PENALVER	7.27	16.91	207	48.92	470	57.08	8307
HUFFINS	7.49	15.57	204	48.72	470	60.62	8300
PLAZIAT	7.82	14.85	204	45.34	490	52.18	8282
MAGNUSSON	7.28	15.52	195	43.78	480	61.10	8274
SMITH	7.47	16.97	195	49.54	500	64.34	8271
MUELLER	7.25	14.69	195	45.90	510	66.10	8253
CHMARA	7.75	14.51	210	42.60	490	54.84	8249

- Auswahl von Zeilen und Spalten, z.B. `d.sport[, 'kugel']`

```
[1] 15.66 13.60 15.82 15.31 16.32 14.01 13.53 14.71 16.91 15.57 14.85 15.52
[13] 16.97 14.69 14.51
```

Dataframes

Klasse `data.frame` — enthält Vektoren gleicher Länge

- Klasse und Dimensionen**

```
data (iris)
class (iris)
dim (iris)
ncol (iris)
nrow (iris)
```

der Iris-Datensatz
'data.frame'
[1] 150 5
[1] 5
[1] 150

- Spalten- und Zeilennamen**

```
names (iris)      [1] 'Sepal.Length' 'Sepal.Width' ... 'Species'
colnames (iris)          dto.
rownames (iris)    [1] 1 2 3 4 5 6 7 ... 150
```

- Konstruktor**

```
daf <- data.frame (x=1:3,
ch=c('shoo','bee','doo'), cpl=rep(2i,3))
```

Dataframes

Attributvektoren gleicher Länge, aber unterschiedlichen Typs

- **Selektion der Attributvektoren**

```
iris$Sepal.Length [1] 5.1 4.9 4.7 4.6 5.0 ... 5.9
iris[[1]] [1] 5.1 4.9 4.7 4.6 5.0 ... 5.9
```

- **Attributvektoren vom Typ 'factor'**

```
iris$Species [1] setosa setosa setosa ... virginica
```

- **Attributnamen als lokale Variable**

```
attach (iris) Namen zuordnen
Sepal.Length [1] 5.1 4.9 4.7 4.6 5.0 ... 5.9
detach (iris) Namen entfernen
```

- **Selektion von Teildatensätzen wie 'matrix'**

iris[,3]	nicht wie iris\$Petal.Length
iris[17,]	das Muster der 17. Zeile
iris[11:20,3:5]	Datensatz mit 10 Zeilen, 3 Merkmalen
iris[3:5]	Vorsicht: Listenselektion!

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Faktoren (Klasse factor)

Speicherökonomische Darstellung *kategorialer* Variablen

- **Faktor \triangleq integercodierter Wertevektor**

```
data (iris); attach (iris); der Iris-Datensatz
print (Species) [1] setosa setosa setosa ... virginica
class (Species) [1] 'factor'
as.vector (Species) [1] 'setosa' 'setosa' ... 'virginica'
unclass (Species) [1] 1 1 1 1 ... 2 2 ... 3 3 3
as.integer (Species) [1] 1 1 1 1 ... 2 2 ... 3 3 3
```

- **Codebuch eines Faktors**

```
levels (Species) [1] 'setosa' 'versicolor' 'virginica'
length (levels (Species)) [1] 3
nlevels (Species) [1] 3
class (levels (Species)) [1] 'character'
```

Faktoren (Klasse factor)

Speicherökonomische Darstellung *kategorialer* Variablen

- **Konstruktor**

```
factor (c(T,T,T,F,F,T)) [1] TRUE TRUE TRUE FALSE FALSE TRUE
factor (c(5,4,7,5,4,7)) [1] 5 4 7 5 4 7
unclass (factor (c(5,4,7,5,4,7))) [1] 2 1 3 2 1 3
factor (c('a','b','b','a')) [1] a b b a
```

- **Typcheck und Konversion**

```
is.factor (Species) [1] TRUE
as.factor (5:8) wie factor (5:8)
```

- **Manipulation des Codebuchs: Reihenfolge**

```
factor (c(T,F,F), levels=c(T,F)) [1] TRUE FALSE FALSE
unclass (factor (c(T,F,F)), levels=c(T,F)) [1] 1 2 2
unclass (factor (c(T,F,F)), levels=c(F,T)) [1] 2 1 1
```

- **Manipulation des Codebuchs: Wertebereich**

```
unclass (factor (3:5)) [1] 1 2 3
unclass (factor (3:5, levels=1:5)) [1] 3 4 5
```

Faktoren $\hat{=}$ Musterklassen

Etikettierte Stichprobe $\hat{=}$ Dataframe ($N \times$ numerical & $1 \times$ factor)

- Klasseninformation als letzte Variable**

<code>iris [[length(iris)]]</code>	der Faktor
<code>iris [length(iris)]</code>	nicht der Faktor
<code>iris [-length(iris)]</code>	die Merkmale
<code>as.matrix (iris [-length(iris)])</code>	die Merkmalmatrix

- Selektion von Teilstichproben**

<code>iris [47:11,]</code>	die Muster 11–47 rückwärts
<code>iris [Species=='setosa',]</code>	die Muster 1–50

- Berechnung klassenweiser Statistiken**

<code>mean (iris [Species=='setosa', 'Petal.Width'])</code>	[1] 0.246
<code>var (iris [Species=='setosa', 'Petal.Width'])</code>	[1] 0.011

- Faktoren und Volkszählung**

<code>table (Species)</code>	[1] 50 50 50
------------------------------	--------------

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Nicht vektorisierte Kontrollstrukturen

Nur für kleine, äußere Schleifen verwendbar !!!

- Einseitige Wertverzweigung**

<code>if (COND) EXPR</code>	logical[1], expression[1]
-----------------------------	---------------------------

- Zweiseitige Wertverzweigung**

<code>if (COND) EXPR.1 else EXPR.2</code>	
---	--

- Gezählte Wiederholung**

<code>for (VAR in SEQ) EXPR</code>	Liste/Vektor 1 \times zu Beginn ausgewertet Weiterschaltung mit <code>next</code>
------------------------------------	--

- Abweisende Wiederholung**

<code>while (COND) EXPR</code>	logical[1], expression[1]
--------------------------------	---------------------------

- Unbedingte Wiederholung**

<code>repeat EXPR</code>	Ausbruch mit <code>break</code>
--------------------------	---------------------------------

Bemerkung

Klammern { und } um zusammengesetzte Ausdrücke nicht vergessen!

Kein Zeilenvorschub vor `else` einfügen (syntaktisch ambig)!

Kontrollstrukturen besitzen einen Wert \Rightarrow letzter Ausdruck in letztem Durchlauf

Vektorisierte Kontrollstrukturen

Fallunterscheidung und komponentenweise Wertverzweigung

- Vektorisierte zweiseitige Wertverzweigung**

<code>x <- ifelse (test, yes, no)</code>	1 \times logical, 2 \times expression drei Vektoren gleicher [...] Länge
---	---

- Fallunterscheidung nach Positionen**

<code>x <- switch (EXPR, exp1, exp2, exp3,)</code>	integer[1]
--	------------

- Fallunterscheidung nach Namen**

<code>x <- switch (EXPR, nam1=exp1, nam2=exp2, nam3=exp3,)</code>	character[1]
---	--------------

- Fallunterscheidung mit Voreinstellung**

(nur Namen; ohne Voreinstellung ist <code>NULL</code> Default)	
--	--

<code>x <- switch (EXPR, nam1=exp1, nam2=exp2,, exp.def)</code>	
---	--

- Fallunterscheidung mit Mehrfachklauseln**

<code>x <- switch (EXPR,, nam1=, nam2=, nam3=exp,)</code>	
--	--

Vektorisierte Iteration über Listen und Vektoren

Iterationsrumpf wird nur **1x** gepasst!

- Mehrfache ($n \times$) Auswertung eines Ausdrucks**

`replicate (n, expr, simplify=TRUE)`

Resultat $\hat{=}$ Liste oder Vektor/Matrix (`simplify=F/T`)

- Funktionsanwendung auf Listenelemente**

`lapply (X, FUN, ...)` die Argumente ... werden der Fkt. `FUN` serviert

Resultat $\hat{=}$ Liste der `FUN(X[[i]])`

- `lapply (list (4,9,16), sqrt)` list (2,3,4)
`lapply (list (4,9,16), '-', 3)` list (1,6,13)

- Funktionsanwendung auf Listen- oder Vektorelemente**

`sapply (X, FUN, ..., simplify=TRUE, USE.NAMES=TRUE)`

Resultat je nach `simplify`; ggf. werden die `X`-Namen eingebunden

- `sapply (1:5, sqrt)` Vektor mit Wurzeln
`sapply (1:5, rep, 3)` (3 \times 5)-Matrix
`sapply (iris[,-5], mean)` Mittelwertvektor

Iterierte Funktionsanwendung mit mehreren Variablen

`mapply (FUN, ..., MoreArgs=NULL)`

- Verallgemeinert sapply**

`mapply (FUN=sqrt, 1:8)` wie `sapply (X=1:8, FUN=sqrt)`

- Funktionen mit zwei und mehr Argumenten**

(alle Argumentvektoren sind von gleicher Länge)

`mapply (FUN='+', 1:4, 4:1)` ergibt [1] 5 5 5 5

`mapply (FUN=rep, 1:4, 4:1)` {{1,1,1,1},{2,2,2},{3,3},{4}}

- Benannte Argumente können adressiert werden**

`mapply (FUN=rep, times=1:4, x=4:1)`

{}{4},{3,3},{2,2,2},{1,1,1,1}}

- Weitere benannte Argumente mit Konstanten belegen**

`mapply (FUN=rep, times=1:4, MoreArgs=list(x=8))`

{}{8},{8,8},{8,8,8},{8,8,8,8}}

Vektorisierte Iteration über mehrdimensionale Felder

`apply (X, MARGIN, FUN, ...)`

- Funktionsanwendung auf Matrixzeilen**

`apply (iris[,-5], MARGIN=1, FUN=max)` 150 Zeilenmaxima

- Funktionsanwendung auf Matrixspalten**

`apply (iris[,-5], MARGIN=2, FUN=mean)` 4 Spaltenmittel

`apply (iris[,-5], MARGIN=2, FUN=range)` (2 \times 4)-Matrix

- Funktionsanwendung auf Matrixelemente**

`apply (iris[,-5], MARGIN=c(1,2), FUN='/', 100)` Umrechnung [cm] in [m]

`apply (iris[,-5], MARGIN=c(1,2), FUN=rep, 3)` (3 \times 150 \times 4)-Kubus

- Hinausrechnen von Statistiken**

`sweep (x, MARGIN=1, STATS=a, FUN=' - ')` subtrahiert a_i von Zeile i

`sweep (x, MARGIN=2, STATS=s, FUN=' / ')` dividiert Spalte j durch s_j

- Spezialwerkzeug: $\mu = 0$ und/oder $\sigma = 1$**

`scale (x, center=TRUE, scale=TRUE)`

Funktionsanwendung auf faktorgruppierte Datenvektoren

`tapply (X, INDEX, FUN=NULL, ..., simplify=TRUE)`

- Klassenweise Mittelwertbildung**

`tapply (iris[[2]], iris[[5]], mean)` nur 1 Faktor

- Auch Wahrheitswerte werden Faktoren**

`z <- runif(100); tapply (z, z>0.5, sum)` 2 Levels

- Warum nicht auch mehrere Faktoren?**

`tapply (iris[[2]], list (iris[[5]], iris[[3]]<5), length)` 2D-Tabelle

- Spezialwerkzeuge für Dataframes**

berechnet Merkmalstatistiken für alle Gruppen

`aggregate (x, by, FUN, ...)` x Datensatz, by Faktorliste

`by (data, INDICES, FUN, ..., simplify=T)` \leadsto by-Objekt

`split (x, f, drop=FALSE, ...)` \leadsto list-Objekt

- Spezialwerkzeug für Vektoren**

`ave (x, ..., FUN=mean)` x Vektor, ... Faktoren

Sonstige Schleifenersatzfunktionen

- **Summenbildung**

`rowSums (x)` entspricht `apply (x, MARGIN=1, FUN=sum)`
`colSums (x)` entspricht `apply (x, MARGIN=2, FUN=sum)`

- **Mittelwertbildung**

`rowMeans (x)` entspricht `apply (x, MARGIN=1, FUN=mean)`
`colMeans (x)` entspricht `apply (x, MARGIN=2, FUN=mean)`

Argument `na.rm=F` zur NA-Feinsteuerung

Argument `dim=1` für höherdimensionale Felder

- **Kumulative Arithmetik**

<code>cumsum (x)</code>	$y_k := \sum_{i=1}^k x_i$
<code>cumprod (x)</code>	$y_k := \prod_{i=1}^k x_i$
<code>cummin (c(3:1,2:0,4:2))</code>	3 2 1 1 1 0 0 0 0
<code>cummax (c(3:1,2:0,4:2))</code>	3 3 3 3 3 3 4 4 4
<code>diff ((0:8)^2)</code>	1 3 5 7 9 11 13 15

- **Äußeres Produkt**, z.B. Vandermonde-Matrix:

`outer (X=z, Y=seq(along=z)-1, FUN="^")` $V_{ij} := z_i^{j-1}$

Vektorisierte Feldkomponentenselektion

Schleifenfreier Zugriff auf Elementesequenz nach Agendamatrix

- **(Konter)diagonale einer Matrix**

`(A <- matrix (1:25, 5, 5))`

`diag (A)`

`A[cbind(1:5,1:5)]`

`A[cbind(5:1,5:1)]`

`A[cbind(1:5,5:1)]`

`A[cbind(5:1,1:5)]`

$$\begin{bmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{bmatrix}$$

1 7 13 19 25

1 7 13 19 25

25 19 13 7 1

21 17 13 9 5

5 9 13 17 21

- **Anwendungsbeispiel: Travelling Florist Problem**

`D <- as.matrix (dist (iris[1:4]))` 150×150 -Distanzmatrix
`p <- sample (nrow (iris))` 65 95 108 114 25 141 126
`path <- cbind (p, c(p[-1],p[1]))` 65,95 95,108 ... 141,126 126,65
`sum (D[path])` 383.8539 [kumulative Distanz]

- **Analoge Vorgehensweise für array-Objekte ...**

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Deklaration und Aufruf von Funktionen

In 'R' sind Funktionen „*Objekte erster Klasse*“

- **Funktionsdeklaration mit formalen Parametern:**

`funcname <- function (arglist) {body}`

- Parameterbezeichner `vname`
- Parameter mit Voreinstellung `vname=default`
- Restparameterliste ...

- **Funktionsaufruf mit aktuellen Parametern:**

`funcname(arglist)`

- Positionelle Übergabe `expr`
- Namentliche Übergabe `vname=expr`
- Restparameterübergabe ...

- **Beispiel:**

```
zeichne <- function (x, y=NULL, title='Grafik', ...) {
  if (is.null (y))
    { y <- x; x <- 1:length(x) }
  plot (y ~ x, main=title, ...)
}

zeichne (sin(1:20), cos(1:20), title='Kreis', col='red')
```

Hilfsmittel zur Funktionsdeklaration

- Konstruktor**

```
function (<>arglst>) <>bodyexpr> Wert ≡ Funktionsobjekt
```

- Rückgabewert (return vs. invisible)**

```
function (x) { x2 <- x^2; sqrt(sum(x2)) } (der letzte Ausdruck)
function (x) return (sqrt(sum(x^2))) (explizit: ± geschwätzig)
function (x) { y <- x%o%x; z <- y%o%y; invisible(z) }
```

- Abbruch, Warnung, Zusicherung**

```
stop ("Halt!", "mich!", "an!")
warning ("Das wird", "teuer!")
stopifnot (is.matrix (S))
```

(Abbruch, Meldung, Position)
(Meldungen werden akkumuliert)
(bei Verstoß Abbruch und Report)

- Benutzerdeklarierte Operatoren**

```
"%xor%" <- function (x,y) x != y
c(T,T,F,F) %xor% c(T,F,T,F) FALSE TRUE TRUE FALSE
```

- Benutzerdeklarierte Zuweisungsoperatoren**

Parametername immer **value** für RHS-Objekt!

```
"plus<-" <- function (x,value) x <- x+value
z <- 1:5; plus(z) <- 10; print(z) 11 12 13 14 15
```

Syntax und Semantik

Vektoren — die elementaren Datentypen

Matrizen — zwei- und mehrdimensionale Felder

Listen — aggregieren Objekte unterschiedlichen Typs

Dataframes — flexible Klasse für Datensätze

Faktoren — eine Klasse für nominale Attribute

Kontrolle — traditionell & vektorisiert

Funktionen — Deklaration & Aufruf

Klassen und Objekte

Informationen über Funktionsobjekte

Nur Psychopathen manipulieren einen Kantorovic-Baum!

- Liste formaler Funktionsparameter**

```
formals (ls) oder formals ("ls")
args (fun)
```

(Argumentliste: Name/Default)
(dto., aber in Textform)

- Funktionsrumpf als 'R'-Sprachobjekt**

```
body (fun)
is.language (body (fun))
```

(Objekt der Klasse **name**, **expression** oder **call**)
TRUE

- Online-Dokumentation abfragen**

```
help (fun) oder ?fun
help.search (pattern) oder ??pattern
apropos (what=<>pattern>)
example (fun)
```

Hilfetext zu Funktion
Hilfetext zu Stichwort
Objektliste mit Treffern
Beispielaufrufe ausführen

Klasse und Datentyp

'R' kam nicht als objektorientierte Sprache auf die Welt

- Atomarer Typ der Komponenten eines Feldes**

```
mode (x)
```

character[1]

- Klasse eines 'R'-Objekts**

```
class (x)
```

character[L]

(1) explizite Klasse:

erste Komponente von **attr(x,'class')**

(2) implizite Klasse:

Matrix/Array; je nach **length(dim(x))**

(3) implizite Klasse:

mode(x) für Vektoren

- Test auf Abstammung von einer Klassenauswahl**

```
inherits (x, what, which=FALSE)
```

logical[1:L]

- Verleihen des Klassenattributs durch Zuweisung**

```
class(x) <- c('myofb', 'lol', 'imho')
```

- Reduktion auf elementaren Typ [...]**

```
y <- unclass (x)
```

Klasse und Datentyp

Beispiele zur Orientierung in einer feindseligen Programmierumgebung

Objekt x	Typ mode(x)	Klasse class(x)	Reduktion class(unclass(x))
NA	logical	logical	logical
883	numeric	numeric	numeric
1618:1648	numeric	integer	integer
1618:1648/17	numeric	numeric	numeric
diag(7)	numeric	matrix	matrix
diag(7)%o%diag(7)	numeric	array	array
iris	list	data.frame	list
iris\$Species	numeric	factor	integer
print	function	function	function
+'	character	character	character

Objektattribute

Attribut \triangleq Name (Zeichenkette) + Wert ('R'-Objekt)

- Abfrage und Änderung

```
attr (x, which="dim") <- 4:5      <- NULL um Attribut zu löschen
```

```
attr (x, which="dim") oder dim (x)    4 5
```

- Alle Attribute auf einmal

```
attributes (x) <- value und attributes (x)
```

- Allgemeine Objektkomposition

```
structure (.Data, ...)           Zusatzargumente in name=value-Form
```

- Standardattribute und ihre Abfrage/Zuweisung

class	dim	dimnames	names	row.names	levels	comment
class()	dim()	dimnames()	names()	row.names()	levels()	comment()
	length()		colnames()	rownames()		
character	integer	character	character	character	character	?

Betrachten von Objektinhalten

Methoden zur textuellen und graphischen Ausgabe

- Textuelle Standardanzeige

```
print (1:3) oder 1:3 oder (x <- 1:3)      [1] 1 2 3
all (print(iris) == iris)                  TRUE
```

- Kompakte, aber erschöpfende Inhaltsangabe

```
str (iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width: num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width: num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species: Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 ...
```

- Semantische Objektzusammenfassung

```
summary (rnorm (1000))                primär für Datenmodellierungsobjekte
Minimum 1st Quant. Median val. Mean val. 3rd Quant. Maximum
-2.856 -0.634 -0.019 -0.026 0.601 2.408
```

- Graphische Standardanzeige

```
plot (iris)
```

Name, Wert und Namensraum

Ein Name ist eine Zeichenkette & ist doch keine Zeichenkette

- Objektname versus Zeichenkette

```
z <- levels(iris$Species); z == "z" FALSE FALSE FALSE
```

- Namensbindung eines 'R'-Objekts verfolgen

```
get ("z")                      "setosa" "versicolor" "virginica"
```

- Neuen Namen an 'R'-Objekt binden

```
assign (x="neu", value=z); neu[3]          "virginica"
```

- Funktionsrümpfe bilden einen Namensraum

```
setx <- function (val) x <- val
x <- "mega-out"; setx (4711); print (x) "mega-out"
```

- Globaler Zuweisungsoperator '<<-'

```
gsetx <- function (val) x <<- val
x <- "mega-out"; gsetx (4711); print (x) 4711
```

Zuweisung — Bindung oder Kopie?

- **Zuweisungsoperator (LHS $\hat{=}$ 'R'-Objekt)**

```
A <- matrix(rnorm(15), 5, 3)      Bindung der RHS an A
c(0,8,15) -> b oder b = c(0,8,15)
```

A <- b kein Problem! (Matrix hingerichtet)

- **Zuweisungsoperator (LHS ist ein 'fun<-'-Aufruf)**

```
A[2,] <- b                           kopiert in zweite Zeile
A[,3] <- b                           dritte Spalte zu lang!
A[6:8] <- b                           Matrix  $\rightsquigarrow$  Vektor
A[] <- b                             Glück gehabt! (3 teilt 15)
```

- **Nutzen klassenerhaltender Zuweisung**

```
C <- matrix(6:1, 2, 3)
sort(C)
C[] <- sort(C); print(C)
```

$$\begin{bmatrix} 6 & 4 & 2 \\ 5 & 3 & 1 \end{bmatrix}$$

1	2	3	4	5	6
[1]	3	5			
[2]	4	6			

Deklaration generischer Methoden

- **Deklaration einer generischen Funktion**

```
foo <- function(x, y, ...) UseMethod('foo', x)
```

- **Delegieren eines generischen Aufrufs**

```
foo(x, ...)                          mit class(x) = c(c1, c2, ..., cn)
initiiert folgende Kette von Delegierungsversuchen:
foo.c1(x, ...)                      die Klasse von x
foo.c2(x, ...)                      eine Oberklasse von x
...
foo.cn(x, ...)                      höchste Oberklasse von x
foo.default(x, ...)                letzte Chance; unbedingt deklarieren !!
```

- **Welche Methoden sind aktuell deklariert?**

```
methods('print')                   alle Methoden print.classname
methods(class='matrix')           alle Methoden fctname.matrix
```

Generische Methoden und polymorpher Aufruf

Generischer Aufruf

```
plot(iris$Species)
```

1. Delegieren an Funktion `plot.factor`?
2. Delegieren an Funktion `plot.integer`?
3. Delegieren an Funktion `plot.default`?

Wir basteln uns eine Graubildklasse

- **Warum? — Darum!**

```
plot(diag(69))
```

`plot.matrix` verwendet nur die Spalten 1 und 2

- **Konstruktor für die BILD-Klasse**

```
BILD <- function(x) {
  x <- matrix(as.integer(cut(x, 256))-1, ncol=ncol(x))
  class(x) <- "BILD"
  invisible(x)}
```

- **Plotmethode für die BILD-Klasse**

```
plot.BILD <- function(x)
  image(x, col=gray(0:255/255))
```

- **Aufruf der Rasterplotmethode**

```
plot.BILD(diag(69)) oder plot(BILD(diag(69)))
```

Zusammenfassung (2)

1. 'R' ist eine **funktionale** Programmiersprache, besitzt aber zahlreiche **Infixoperatoren** als syntaktischen Zucker.
2. Atomare 'R'-Datentypen sind die **Vektoren** (numerisch, logisch, komplex, Strings); dazu gibt es flexible **Indexierungsmechanismen** und **komponentenweise** Arithmetik.
3. **Matrizen** und mehrdimensionale Felder sind als Vektoren mit **Dimensionsattribut** realisiert; Unterstützung der **linearen Algebra**.
4. **Listen** sind hierarchische Verbundobjekte. Wie auch die Felder unterstützen sie **Komponentennamen**.
5. **Datensätze** enthalten numerische, aber auch kategoriale (**Faktoren**) Merkmale.
6. Als **interpretative** Sprache gebietet 'R' den Gebrauch **vektorisierter** statt traditioneller **Kontrollstrukturen** („Schleifen“).
7. In 'R' sind Funktionen **Datenobjekte** erster Klasse.
8. 'R' bietet rudimentäre **Objektorientierung** durch **Klassenattribute** und **polymorphen** Funktionsaufruf (Argument #1).

WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 19. Februar 2018

Teil III

Text, Datensätze und Grafik in 

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

(Vektoren von) Zeichenketten

Zeichen in 'R' \triangleq Zeichenketten der Länge 1

- **Konstruktion und Konversion**

character(5)	"" "" "" "" "
as.character(5:1)	"5" "4" "3" "2" "1"
LETTERS [-(4:24)]	"A" "B" "C" "Y" "Z"
DDR <- c ("Deutsche", "Demokratische", "Republik")	

- **Projektion und Transformation**

nchar (DDR)	8 13 8
c (tolower(DDR)[1], toupper(DDR)[3])	"deutsche" "REPUBLIK"

- **Ersetzung von Zeichen und Ketten**

chartr (old=DDR[1], new=DDR[3], x=DDR[2])	"Rkmokrauiblik"
sub ("t", "TEE", DDR[1])	"DeuTEEsche"
sub (".", "?", DDR[1])	"?eutsche"
gsub (".", "?", DDR[1])	"?????????"

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Suchen, Zerschneiden und Verketten

- Suche nach Mustervorkommen**

```
grep ("Tisch", DDR)           integer(0)
grep ("Tisch", DDR, ignore.case=TRUE) 2
```

- Zerschneiden von Ketten**

```
strsplit (DDR[1], split="e")      "D" "utsch"
strsplit (DDR[1], split="")       "D" "e" "u" "t" "s" "c" "h"
substr (x=DDR, start=3, stop=5)   "uts" "mok" "pub"
abbreviate (DDR, minlen=4)       "Dtsc" "Dmkr" "Rpbl"
```

- Verkettung (Konkatenation)**

```
paste ("hello", "world", "!")    "hello world !"
paste ("TOP", 1:3)               "TOP 1" "TOP 2" "TOP 3"
paste ("BMW ", seq(1,7,by=2), 30, "i", sep="")
                                "BMW 130i" "BMW 330i" "BMW 530i" "BMW 730i"
paste (letters[1:8], 4, sep="", collapse=".")
                                "a4·b4·c4·d4·e4·f4·g4·h4"
```

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Partielles String Matching

```
pmatch (x, table, nomatch=NA, duplicates.ok=FALSE)
```

- Index kompletter/partieller Vorkommen**

```
average <- c("mean", "median", "mode")
pmatch (c("mode", "med", "foo"), average) 3 2 NA
```

- Partielle Treffer müssen eindeutig sein!**

```
pmatch (c("me", "mo", ""), average) NA 3 NA
pmatch (c("me", "mo", ""), average, nomatch=0) 0 3 0
```

- Getroffene Einträge werden deaktiviert!**

```
pmatch (c("medi", "med"), average) 2 NA
pmatch (c("medi", "med"), average, dup=TRUE) 2 2
```

- Komplette Treffer zuerst!**

```
pmatch (c("medi", "median"), average) NA 2
```

- Komplette Treffer werden immer akzeptiert!**

```
pmatch(c("L", "L"), c("LOL", "L"), dup=FALSE) 2 1
pmatch(c("L", "L"), c("LOL", "L"), dup=TRUE) 2 2
```

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Uunausgewertete Ausdrücke als Sprachobjekte

Klassen `expression`, `name` und `call`

- Programmtext ~ Ausdruck**

```
parse (text="1:3+4; iris; NA") -> ex
print (ex)                  expression (1:3+4, iris, NA)
class (ex)                   "expression"
parse (file="prog.R")        Programmtext von Datei lesen
```

- Der Kantorovic-Baum eines Ausdrucks**

```
sapply (ex, class)          "call" "name" "logical"
sapply (ex[[1]], class)     "name" "call" "numeric"
sapply (ex[[1]][[2]], class) "name" "numeric" "numeric"
```

- Auswerten eines (unausgewerteten) Ausdrucks**

```
eval (ex)                   NA
eval (ex[[1]])              5 6 7
eval (ex[[1]][[1]])         function (e1, e2) .Primitive("+")
```

- Ausdruck ~ Programmtext**

```
supply (ex, deparse)        "1:17 + 4" "iris" "NA"
```

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

Datensätze in Mustererkennung und Data Mining

Matrix $\hat{=}$ Objekte (Fälle, Muster) \times Attribute (Variable, Merkmale)

Beispieldatensatz

	vorbestraft	Partei	Abinote	Geburt	Spenden
Angela	F	CDU	gut	1954	$345 \cdot 10^3$
Guido	F	FDP	ausreichend	1961	$137 \cdot 10^3$
Roland	T	CDU	gut	1958	$3.6 \cdot 10^6$
Gregor	F	PDS	sehr gut	1948	NA
Linus	F	Pirat	NA	1969	0
Bill	F	Rep	mangelhaft	1955	$-4.2 \cdot 10^9$
Roman	T	1933	...
:	:	:	:	:	:
:	:	:	:	:	:

Range	$\{T, F\}$	$\{\pi_1, \dots, \pi_9\}$	$\{\nu_1, \dots, \nu_5\}$	$\mathbb{Z} \subset \mathbb{R}$	\mathbb{R}
Skala	nominal	nominal	ordinal	relativ	absolut
'R'-Typ	factor	factor	ordered	numeric	numeric

Erzeugen von Datensätzen

data.frame $\hat{=}$ Liste von 'R'-Vektoren gleicher Länge

- Konstruktoraufruf \sim Objekt der Klasse data.frame

data.frame(..., row.names=NULL)

Parameter: $\begin{cases} \text{Vektoren} \\ \text{Matrizen} \end{cases}$ $\begin{cases} \text{mit} \\ \text{ohne} \end{cases}$ Namen; Zeilennamen: $\begin{cases} \text{char(n)} \text{ explizit} \\ \text{int(1)} \text{ implizit} \end{cases}$

- Matrixkonversion (Spalten=Merkmale)

as.data.frame(x=diag(5), row.names=LETTERS[1:5])

- Laden verfügbarer Datensätze

data()

listet alle Datensätze

data(package="MASS")

listet MASS-Datensätze

data(eagles, package="MASS")

macht eagles sichtbar

- Lesen einer Datensatzdatei

read.table, load, source, ... (je nach Format)

Datensätze laden — auch über das Internet

- Als Dateinamen werden auch URLs akzeptiert:

```
myurl <- 'http://stat.ethz.ch/Teaching/Datasets/NDK/sport.dat'
sport <- read.table(myurl)
```

- Datensätze mit benannten Mustern & Merkmalen:

	weit	kugel	hoch	disc	stab	speer	punkte
OBRIEN	7.57	15.66	207	48.78	500	66.90	8824
BUSEMANN	8.07	13.60	204	45.04	480	66.86	8706
DVORAK	7.60	15.82	198	46.28	470	70.16	8664
FRITZ	7.77	15.31	204	49.84	510	65.70	8644
HAMALAINEN	7.48	16.32	198	49.62	500	57.66	8613
NOOL	7.88	14.01	201	42.98	540	65.48	8543
ZMELIK	7.64	13.53	195	43.44	540	67.20	8422
GANIYEV	7.61	14.71	213	44.86	520	53.70	8318
PENALVER	7.27	16.91	207	48.92	470	57.08	8307
HUFFINS	7.49	15.57	204	48.72	470	60.62	8300
PLAZIAT	7.82	14.85	204	45.34	490	52.18	8282
MAGNUSSON	7.28	15.52	195	43.78	480	61.10	8274
SMITH	7.47	16.97	195	49.54	500	64.34	8271
MUELLER	7.25	14.69	195	45.90	510	66.10	8253
CHMARA	7.75	14.51	210	42.60	490	54.84	8249

- Auswahl von Zeilen und Spalten, z.B. sport[, 'kugel']

```
[1] 15.66 13.60 15.82 15.31 16.32 14.01 13.53 14.71 16.91 15.57 14.85 15.52
[13] 16.97 14.69 14.51
```

Datensätze inspizieren

- Matrixeigenschaften

names(sport)

"weit" "kugel" ... "punkte"

rownames(sport)

"OBRIEN" "BUSEMANN" ... "CHMARA"

sport['FRITZ', 'speer']

65.7

sport[m,]

Datensatz, kein Vektor!!

- Listeneigenschaften

length(sport) == ncol(sport)

TRUE

sport[3] versus sport[[3]]

Datensatz oder Vektor ?!

- Attribute in den Namensraum transportieren

attach(sport); mean(kugel); detach(sport)

15.2

with(sport, mean(kugel))

dieselbe Wirkung

- Komfortable Ausschnittbildung 1

subset(x=sport, subset=hoch>=200, select=weit:hoch)

Zeilenwahl (logical) und Spaltenwahl (symbol. Indexausdruck)

Zoo der E/A-Funktionen in R



„Wer lesen und schreiben kann, ist klar im Vorteil!“

Zuständigkeit

'R'-Objekte
'R'-Klassen
'R'-Atome

Logische Darstellung

Typbezogenes Datenformat
Programmtext ('R')
Objektserialisierung

Abstraktionsebene

Systemschnittstelle
Benutzerschnittstelle

Physische Darstellung

Datei — Strom
ASCII — Binärformat
Redundant — Komprimiert

E/A für elementaren Freitext

Funktionen cat (... , file, sep) und scan (file, what)

- **Textausgabe für Atome und Felder** (Standardausgabe)
cat (1:3, 'Wort', matrix(1:4,2)) 1 2 3 Wort 1 2 3 4

- **Ausgabe mit Trennmuster**

cat (LETTERS[1:5], 'A', sep='=>') A=>B=>C=>D=>E=>A

- **Ausgabe in Datei**

cat (1:4, '\n', letters[1:3], '\n', file='foo.txt')

- **Eingabe typgleicher Werte von Datei**

scan ('foo.txt') Fehler in scan() - erwartete 'a real', bekam 'a'
scan ('foo.txt', character(0)) "1" "2" "3" "4" "a" "b" "c"
scan ('foo.txt', what='xy', nmax=4) "1" "2" "3" "4"
scan ('foo.txt', what=numeric(0), nmax=4) 1 2 3 4

- **Formatierte Eingabe (Zeitkiller!)**

lst <- scan ('foo.txt', what=list (0,0,0,0,'',''))
sapply (lst, length) 1 1 1 1 1 1 1

E/A für Datensätze (und Matrizen)

Funktionen write.table und read.table

- **Formatierte Ausgabe von Datensätzen**

write.table (x, file="", append=FALSE)
Konversion, Standardausgabe, Dateibeginn as.data.frame (x)

- **Formatierte Eingabe von Datensätzen**

read.table (file, header=FALSE, sep=',', dec='.', as.is=, ...)
liefert data.frame-Objekt (Spaltennamen?, Zeilennamen?, Layoutdetails?)
seltener im Einsatz:

- **Formatierte Ausgabe von Matrizen** (alle Typen)

write (x, file, ncolumns, append=FALSE, sep=' ')
vorzugsweise ncolumns=ncol(x), denn:

- **Formatierte Eingabe von Matrizen** (als typhomogener Datensatz)

as.data.frame (read.table (file, header=TRUE))
Standardzeilen- und spaltennamen: 1 2 ... m und V1 V2 ... Vn

Transparente E/A für beliebige 'R'-Objekte

Funktionen save und load speichern Objektstruktur & Objektbestandteile

- **Automatische Serialisierbarkeit aller 'R'-Objekte**

A <- matrix (rnorm(1000), 25, 40)
B <- list (name=c('Amy', 'W.'), age.max=27, alive=F)
C <- function (m, c=3e8) E <- m * c^2

- **Gemeinsames Abspeichern auf Datei**

save (A, "B", C, file="foo.rda")
oder alternativ:
save (list=list ('A', 'B', 'C'), file="foo.rda")

- **Gemeinsames Rückladen von Datei**

z <- load (file="foo.rda")
print (z) "A" "B" "C"
ls () unter anderem: A B C

- **Was passiert am Sitzungsende?** (nach quit('yes'))

save (list=ls (all=TRUE), file=".RData")

Operationale E/A für beliebige 'R'-Objekte

Funktionen `dump` und `source` speichern Bauplan / Herstellungsprozeß des Objekts

- **Objektrepräsentation** \triangleq 'R'-Programmtext

AUS: Name \rightsquigarrow Ausdruck \rightsquigarrow Text

EIN: Text \rightsquigarrow Ausdruck \rightsquigarrow Eval \rightsquigarrow Binden

- **Gemeinsames Abspeichern auf Datei**

`dump (list, file="dumpdata.R", append=FALSE, ...)`

Objektnamen als Liste von Zeichenketten

- **Wiederherstellen aller Objekte**

`source (file, local=FALSE, ...)`

Standardumgebung (Namensraum) ist der globale Arbeitsbereich

- **Basisfunktionen** (Einzelobjekte, ohne Namensbindung)

`dput (x=function(x) sqrt(sum(x^2)), file="foo.R")`

`euno <- dget (file="foo.R")`

`euno (3:4) == 5`

TRUE

E/A-Systemschnittstelle

Datenströme, Atome, Serialisierung, Bytevektoren

Datenströme

Klasse `connection`

Datei `file()`

Adresse `url()`

Quelle/Senke `pipe()`

Filter `fifo()`

`open(), close(), flush()`.

Bytevektoren

Atomarer Typ `raw`

Adreßeinheit $\hat{=}$ Byte

Binärrepräsentation der E/A
(`connection=NULL`)

E/A für atomare Vektoren

`readBin()` `writeBin()`

`readChar()` `writeChar()`

`readLines()` `writeLines()`

E/A durch Serialisierung

`serialize()` `saveRDS()`

`unserialize()` `readRDS()`

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

R-Programmpakete

'R'-Implementierung

Primitive Funktionen (unzerlegbar)

Eingebaute Funktionen (C++-Bibliotheken)

Deklarierte Funktionen ('R'-Objekte)

Funktionsumfang

Systemkern (alle primitiven Fkt.)

'R'-Basispaket (deklarierte Fkt.)

Standarddistribution (aktuell 27 Pakete)

CRAN Zusatzpakete (>500 Pakete)

Paketbestandteile

Funktionsdeklarationen

Daten(sätze)

C++-Bibliothek

Handbuchseiten, Vignetten

Paketnutzung

(Bauen & Pflegen)

Installation (Rechner)

Laden (Sitzung)

Aufruf (Fkt.) & Bindung (Daten)

Programmpakete der Standarddistribution

Standort: /usr/lib64/R/library o.ä.

base	The R Base Package
boot	Bootstrap Functions (originally by Angelo Canty for S)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
compiler	The R Compiler Package
datasets	The R Datasets Package
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase
graphics	The R Graphics Package
grDevices	The R Graphics Devices and Support for Colours and Fonts
grid	The Grid Graphics Package
KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)
lattice	Lattice Graphics
MASS	Support Functions and Datasets for Venables and Ripley's MASS
Matrix	Sparse and Dense Matrix Classes and Methods
methods	Formal Methods and Classes
mgcv	GAMs with GCV/AIC/REML smoothness estimation and GAMMs by PQL
nlme	Linear and Nonlinear Mixed Effects Models
nnet	Feed-forward Neural Networks and Multinomial Log-Linear Models
rpart	Recursive Partitioning
spatial	Functions for Kriging and Point Pattern Analysis
splines	Regression Spline Functions and Classes
stats	The R Stats Package
stats4	Statistical Functions using S4 Classes
survival	Survival analysis, including penalised likelihood.
tcltk	Tcl/Tk Interface
tools	Tools for Package Development
utils	The R Utils Package

CRAN Task Views

Standort: http://cran.r-project.org/

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis
gR	gRaphical Models in R

Pakete installieren & laden

- **Installiertes Paket laden**

```
library (package='MASS') oder require (package='MASS')
```

Paket als Name oder Zeichenkette

- **Paketinformationen**

```
library () listet installierte Pakete
library (help='MASS') Details zu genanntem Paket
```

- **Paket installieren (Internetzugang!)**

```
install.packages (pkgs) Paketnamen in Zeichenkettenvektor
update.packages () spielt aktuelle Paketversionen ein
```

- **Alle Pakete einer Anwendungsklasse „task view“**

```
install.views (views="Econometrics")
update.views (views="Econometrics")
```

Vorher muß Paket `ctv` installiert und geladen werden!

```
available.views () listet verfügbare TVs
```

Zugriff auf Komponenten geladener Pakete

- **Sichtbarkeit in der aktuellen Arbeitsumgebung**

- alle Hilfetexte
- die wichtigsten Funktionsobjekte
- keine Datenobjekte

- **Datensätze ruhen in den Installationsdateien**

```
data (... , list=character() , package=NULL , ...)
```

Vier Datenformate können automatisch gelesen werden:

'R'-Quelltext
Serielles Format
Datensatz-Tabelle
Comma-Separated Values

source() *.{R,r}
load() *.{RData,rda}
read.table() *.{tab,txt,TXT}
read.csv() *.{csv,CSV}

R-Paket rimage

Lesen von JPEG-Dateien (und Manipulieren von Bildobjekten)

- Laden des rimage-Pakets

```
library (rimage)
help (package='rimage')
```

- Einlesen der JPEG-Datei in ein imagematrix-Objekt

```
imo <- read.jpeg ('Rlogo.jpg')
```

- Spezifische Methoden für print und plot

```
print (imo)           size: 77 x 101, type: rgb
plot (imo, main='Das R-Logo')
```

- Grauwertbilder sind Matrizen

```
class (imo); attr (imo,'type') [1] 'imagematrix' [1] 'grey'
dim (imo)                      [1] 77 101
```

Bildpunkte manipulieren mit `imo[i,j]`

- Farbbilder

```
class (imo); attr (imo,'type') [1] 'imagematrix' [1] 'rgb'
dim (imo)                      [1] 77 101 3
```

Bildpunkte manipulieren mit `imo[i,j,channel]`

R-Paket rimage

Schreiben von JPEG-Dateien (?!)

Problem

Beim Schreiben einer Bildmatrix (oder RGB-Würfel) als JPEG-Datei unterstützt uns das Paket leider nicht!



- Abspeichern auf Datei als imagematrix-Objekt
`save (imo, file='bild.rda')`

- Hardwareplattformunabhängiges Datenformat
`save (imo, file='bild.rda', ascii=TRUE)`

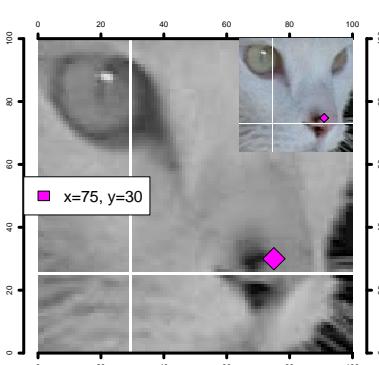
- Exakte Kompression der Bilddaten
`save (imo, file='bild.rda', compress=TRUE)`

- Wiedereinlesen der (Bild-)Daten
`load (file='bild.rda')`

Auch unter den Einstellungen `ascii=F,compress=T` ist die `.rda`-Datei viel größer als die entsprechende JPEG-Datei.
Allerdings stellt sie auch eine **exakte** Repräsentation dar.

R-Paket rimage

Beispiel zu Bildgeometrie, Farbe und Grauwert



```
require (rimage)
dog <- read.jpeg (
  system.file (
    "images", "cat.jpg",
    package="rimage"))
dog <- imagematrix (dog
  [200+1:100,230+1:100,])
dog[75,] <- 1; dog[,30,] <- 1

layout (matrix (c(1,1,2,1), 2, 2))

plot (rgb2grey (dog))
for (i in 1:4)
  axis (side=i, lwd=5)
points (75, 30,
  pch=23, bg="magenta", cex=5)
legend ("left", legend="x=75, y=30",
  cex=2, bg="white", fill="magenta")

plot (dog)
points (75, 30, pch=23, bg="magenta", cex=2)
```

Bildgeometrie

der Klasse 'imagematrix':

x=75 Zeile 75 ⚡
y=30 Spalte 30 ⚡

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

Computergrafische Datenvisualisierung

Explorative Data Analysis: [data \mapsto image \mapsto perception \mapsto HCI]*

Zweck der Visualisierung

Gewinne erste Übersicht
Erzähle eine Geschichte
Generiere eine Hypothese
Überprüfe ein Modell

Mittel der Visualisierung

Punkte, Linien
Koordinatensystem, Achsen
Zahlenwerte, Symbole, Text
Farbe, Schattierung

Fakt

Humans are good at discerning subtle patterns that are really there, but equally so at imagining them when they are altogether absent.
„Contact“ (Carl Sagan, 1986)

Mehr Informationen

Skript „Maschinelles Lernen & Data Mining“, Kapitel 3.8 (Visualisierung:EDA)
<http://www.minet.uni-jena.de/fakultaet/schukat/ML/Scriptum/lect03-visual.pdf>

Grafikproduktion in

... die ersten vier Laufschleifen ...

- **Wohin damit?** (... wenn nicht auf den Bildschirm)

`pdf (file="Rplots.pdf", width=7, height=7, ...)` [inch]

Alternativ: `postscript()`, `jpeg()`, `tiff()`, `svg()`, `xfig()`, `pictex()`, `X11()`.

- **Leinwandaufteilung** („canvas“, z.B. 2×3  

`layout (matrix (1:6,2,3))` oder `par (mfcol=c(2,3))`

- **Primärfunktion** (Koordinatensystem et al.)

`plot (x, sin(x), type='b')`

Alternativ: `boxplot()`, `barplot()`, `hist()`, `contour()`, `persp()`

- **Sekundärfunktionen** (Punkte, Linien, Symbole, Texte)

`points()`, `lines()`, `polygon()`, `segments()`, ... Daten

`abline()`, `rect()`, `rug()`, `symbols()`, `text()`, ... Markierung

`legend()`, `axis()`, `title()`, `mtext()`, ... Beschriftung

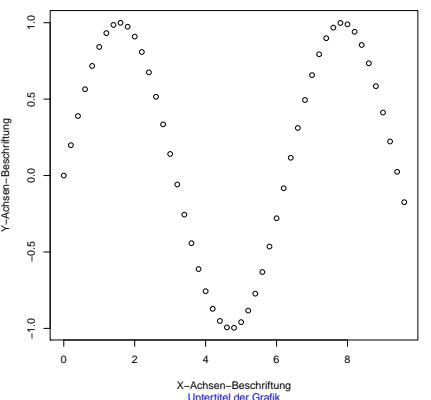
- **Grafikparameter** (siehe `?par`)

1. Default · 2. Justieren: `par (...)` · 3. Übergeben: `plot (...)`

Beschriftungsparameter und -funktionen

Parameter `main`, `sub`, `xlab`, `ylab` · Sekundärfunktion `title()`

Titel der Grafik



```
plot (x <- 0:48/5, sin(x),
      main="Titel der Grafik",
      sub="Untertitel der Grafik",
      xlab="X-Achsen-Beschriftung",
      ylab="Y-Achsen-Beschriftung",
      cex.main=3, col.sub="blue")
```

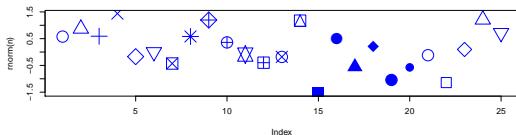
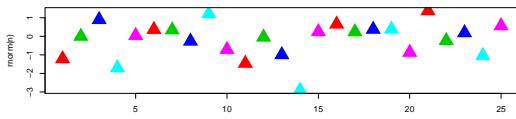
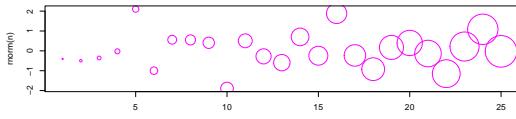
```
# DIESELBE AUSGABE BEWIRKT AUCH ...
plot (x <- 0:48/5, sin(x),
      xlab="", ylab="")
title (main="Titel der Grafik",
       sub="Untertitel der Grafik",
       xlab="X-Achsen-Beschriftung",
       ylab="Y-Achsen-Beschriftung",
       cex.main=3, col.sub="blue")
```

Parameter für Farbe und (relative) Schriftgröße

`{col.main}`, `{col.sub}`, `{col.xlab}`, `{col.ylab}`, `{cex.xlab}` und `{cex.ylab}` und `{cex}` (Punkt/Linie).

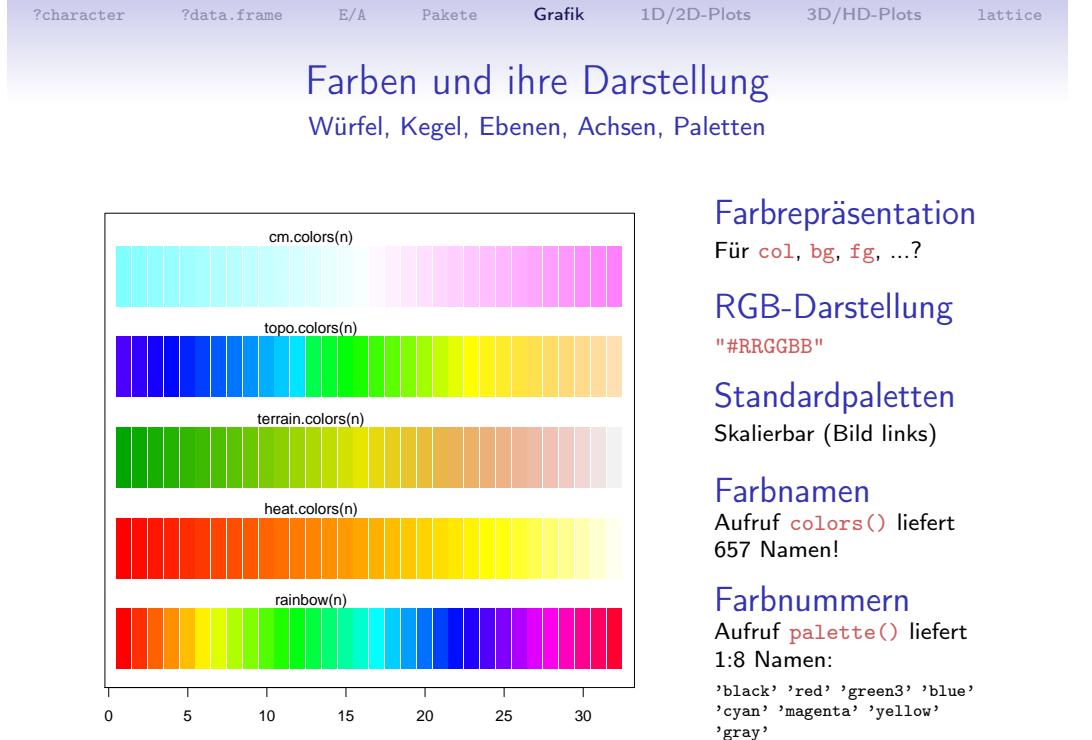
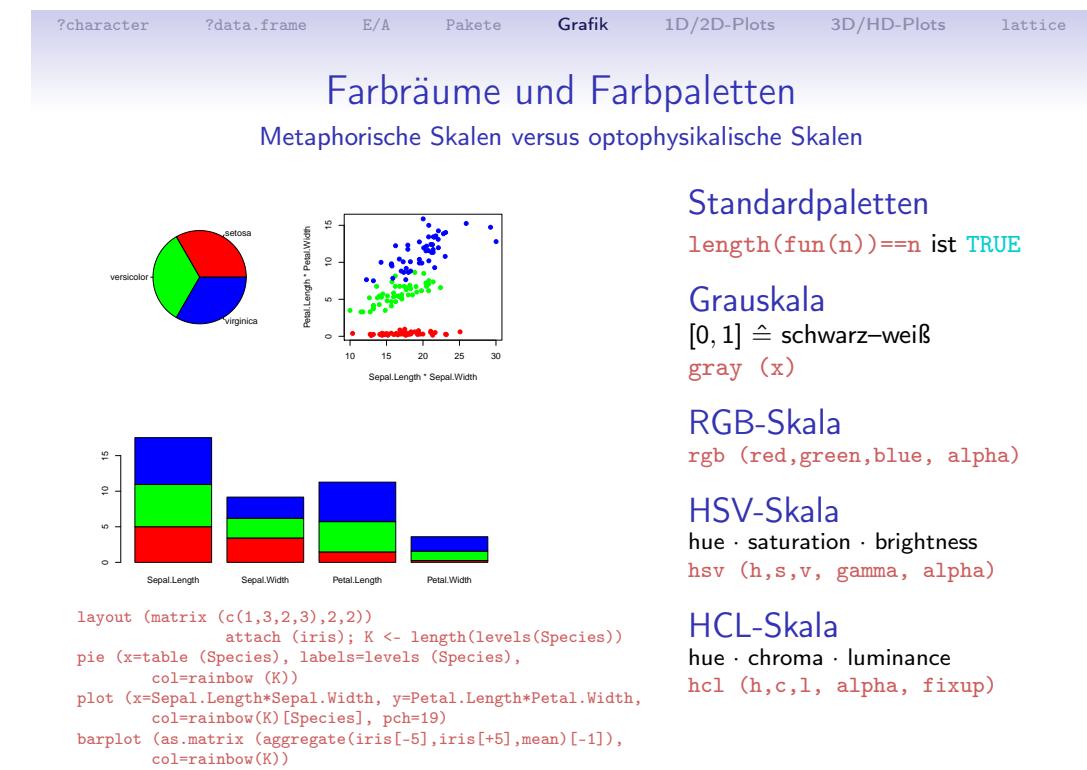
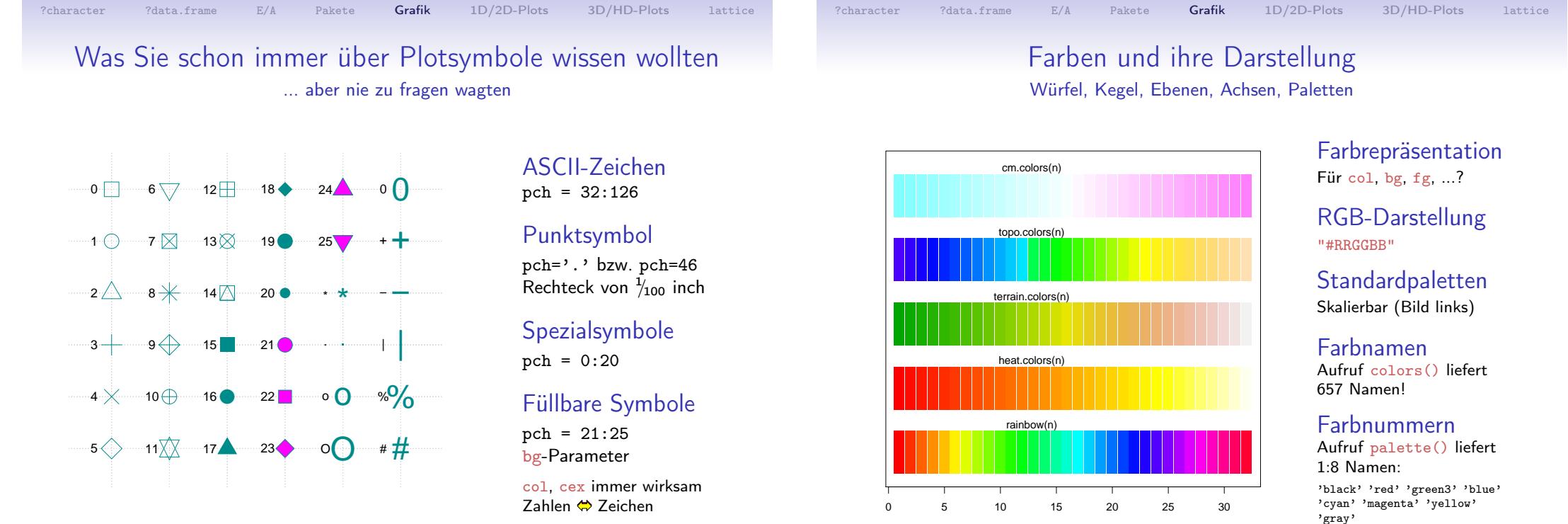
Punktweise Grafikattribute

`cex = Größe` · `col = Farbe` · `pch = Symbol`



```
layout (1:3); n=25
plot (rnorm(n),
      col="magenta",
      cex=1:n/3)
plot (rnorm(n),
      cex=3,
      pch=17,
      col=1+1:5)
plot (rnorm(n),
      cex=3,
      col="blue",
      pch=1:n)
```

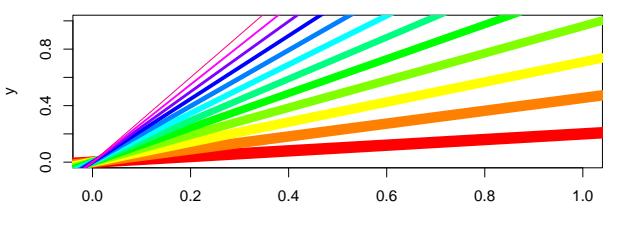
Wirkung  Länge
 = 1 \rightsquigarrow global
 = n \rightsquigarrow lokal
 < n \rightsquigarrow repetitiv



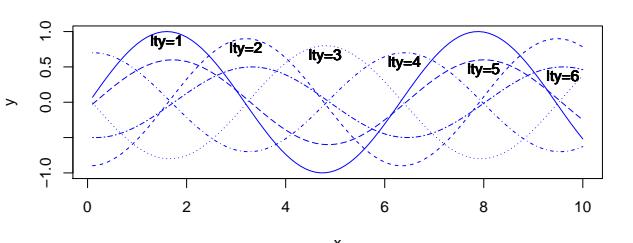
Linienattribute

Linientyp lty, Liniendicke lwd und Farbe col

Linienfarbe & Liniendicke



Linientyp



Liniendicke

$lwd \in \mathbb{R}_0^+$
(Standard=1)

Linientyp

lty ist Zahlencode,
Schlüsselwort oder
Hexasequenz

0	"blank"	"_"
1	"solid"	"11"
2	"dashed"	"44"
3	"dotted"	"13"
4	"dotdash"	"1343"
5	"longdash"	"73"
6	"twodash"	"2262"
-	"_"	"AFFE"

Textinschrift und Abbildungslegende

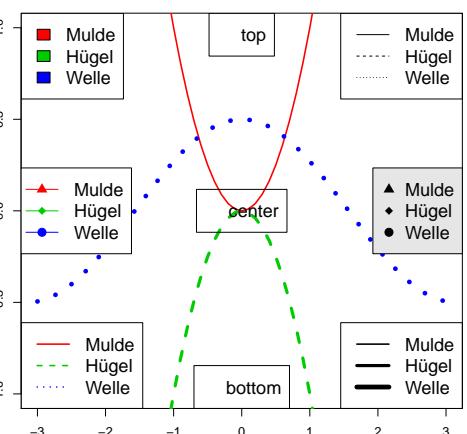
legend (x, y=NULL, legend, fill=NULL, col, lty, lwd, pch, bg, ...)

```
x <- seq (-3,+3,0.1)
txt <- c("Mulde","Hügel","Welle")
Y <- list (x^2, -x^2, cos(x)/2)
clr <- 1:3+1
typ <- 1:3
wid <- 1:3*2
cha <- 1:3+16

plot (x, sin(x),
      type="n", xlab="", ylab="")
for (i in seq(along=Y))
  lines (x, Y[[i]],
         col=clr[i], lty=typ[i], lwd=wid[i])
par (cex=1.4)

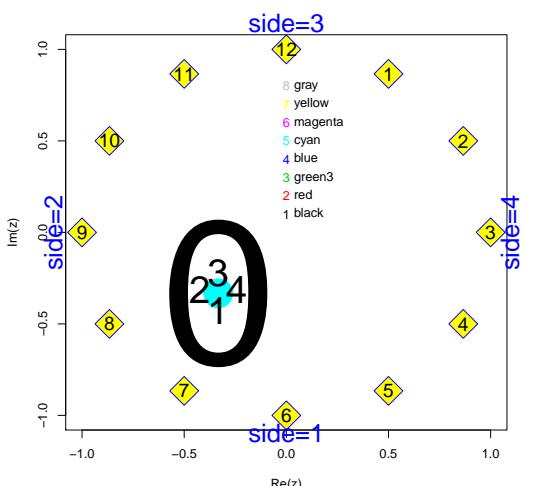
legend ("topleft", legend=txt, fill=clr)
legend ("topright", legend=txt, lty=typ)
legend ("bottomright", legend=txt, lwd=wid)
legend ("bottomleft", legend=txt,
       lwd=2, lty=typ, col=clr)
legend ("right", legend=txt,
       pch=cha, bg=gray(.9))
legend ("left", legend=txt,
       pch=cha, col=clr, lwd=1, bg=gray(1))

for (pos in c("top","bottom","center"))
  legend (pos, legend=pos)
```



Textinschrift und Abbildungslegende

Funktionen $\begin{cases} \text{text (x, y=NULL, labels, pos, cex, ...)} \\ \text{mtext (text, side=3, cex, col, ...)} \end{cases}$



```
z <- exp (2i*pi/12 * 1:12)
plot (Re(z), Im(z),
      pch=23, cex=4,
      col="blue", bg="yellow")
text (z,
      labels=(12:1+1)%/%12+1, cex=1.5)

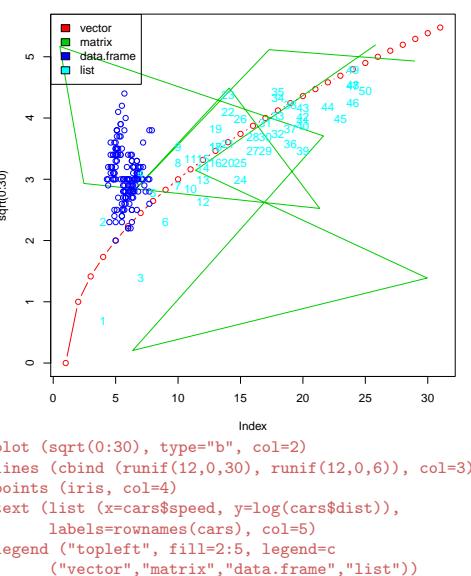
text (x=0, y=1:8/10,
      labels=1:8, col=palette())
text (x=0, y=1:8/10,
      labels=palette(), pos=4)

points (x=-1/3, y=-1/3,
        pch=19, cex=5, col="cyan")
text (x=-1/3, y=-1/3,
      labels=1:4, pos=1:4, cex=3)
text (x=-1/3, y=-1/3,
      labels=0, cex=16)

mtext (
  text=paste ("side",1:4,sep=""),
  side=1:4, col="blue", cex=2)
```

Formulierungsvarianten für 2D-Koordinaten

xy.coords (x, y=NULL, xlab=, ylab=, log=, recycle=FALSE)



```
plot (sqrt(0:30), type="b", col=2)
lines (cbind (runif(12,0,30), runif(12,0,6)), col=3)
points (iris, col=4)
text (list (x=cars$speed, y=log(cars$dist)),
      labels=rownames(cars), col=5)
legend ("topleft", fill=2:5, legend=c
("vector","matrix","data.frame","list"))
```

Schnittstelle x, y-Args

xy.coords()-Aufruf liefert
Liste mit Vektoren x, y und
Zeichenketten xlab, ylab.

Kritischer Fall y=NULL

vector:	Index 1:n und x
complex:	Re(x) und Im(x)
matrix:	Spalten 1,2
data.frame:	Spalten 1,2
list:	x\$x und x\$y
ts:	time(x) und x
formula:	Form yvar ~ xvar

Standardfall recycle=F

Die Argumente x und y müssen
gleiche Länge besitzen!

Mathematik-Layout im L^AT_EX-Stil

'R'-Objekte der Klasse **expression** als Textargumente

Arithmetic Operators	Relations	Juxtaposition
$x + y$	$x + y$	$x == y$
$x - y$	$x != y$	$x = y$
$x * y$	$x < y$	$x < y$
x/y	$x <= y$	$x \leq y$
$x \%+-% y$	$x \pm y$	$x > y$
$x \%/% y$	$x \geq y$	$x \geq y$
$x \%*% y$	$x \sim y$	$x \approx y$
$x \%.% y$	$x \equiv y$	$x \equiv y$
$-x$	$x == y$	$x \equiv y$
$+x$	$x \prop y$	$x \approx y$

Sub/Superscripts
$x[i]$
x^2
Radicals
\sqrt{x}
$\sqrt[n]{x}$
Ellipsis
$list(x[1], \dots, x[n])$
$x[1] + \dots + x[n]$
$list(x[1], \dots, x[n])$
$x[1] + \dots + x[n]$

Set Relations	Arrows
$x \%subset% y$	$x \subset y$
$x \%subseteq% y$	$x \subseteq y$
$x \%supset% y$	$x \supset y$
$x \%supseteq% y$	$x \supseteq y$
$x \%notsubset% y$	$x \not\subset y$
$x \%in% y$	$x \in y$
$x \%notin% y$	$x \notin y$
	$x \%lt% y$
	$x \%gt% y$
	$x \%leq% y$
	$x \%geq% y$
	$x \%lneq% y$
	$x \%gneq% y$
	$x \%ll% y$
	$x \%gg% y$

Accents
\hat{x}
\tilde{x}
\bar{x}
\overline{xy}
\widehat{xy}
\widetilde{xy}

Mathematik-Layout im L^AT_EX-Stil

Hilfeseiten mit Kommando **?plotmath**

Symbolic Names	
Alpha – Omega	$\alpha - \Omega$
alpha – omega	$\alpha - \omega$
phi1 + sigma1	$\phi + \varsigma$
Upsilon1	Υ
infinity	∞
32 * degree	32°
60 * minute	$60'$
30 * second	$30''$

Style
$displaystyle(x)$
$textstyle(x)$
$scriptstyle(x)$
$scriptscriptstyle(x)$
Spacing
$x \sim y$
$x + phantom(0) + y$
$x + over(1, phantom(0))$
Fractions
$frac(x, y)$
$over(x, y)$
$atop(x, y)$

Big Operators
$sum(x[i], i = 1, n)$
$prod(plain(P)(X == x), x)$
$integral(f(x) * dx, a, b)$
$union(A[i], i == 1, n)$
$intersect(A[i], i == 1, n)$
$lim(f(x), x \%-%> 0)$
$min(g(x), x \geq 0)$
$inf(S)$
$sup(S)$

Grouping
$(x + y) * z$
$x^y + z$
x^{y+z}
x^{y^z}
$group("(", list(a, b), ")")$
$bgroup("(", atop(x, y), ")")$
$group(ceil, x, rceil)$
$group(floor, x, rfloor)$
$group(" ", x, " ")$

Mathematik-Layout im L^AT_EX-Stil

Hilfeseiten mit Kommando **?plotmath**

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

Histogramme — Ereignishäufigkeiten in Schubladen

`hist (x, breaks, freq, density, angle, x/ylim, axes, plot, labels, ...)`

Histogram of log(USairpollution\$popul)

Frequency

4 5 6 7 8

2 2 5 6 8 2 1 1

Histogram of rnorm(1000)

Frequency

-3 -2 -1 0 1 2 3

hist (log (USairpollution\$popul), col="blue", labels=TRUE, xlab="")
hist (rnorm (1000), col="red", breaks=32, xlab="")

Schubladengeometrie
± äquidistante Grenzen

Parameter 'breaks'
Grenzen numeric(k)
Anzahl integer(1)
Methode function
Strategie character(1)
Sturges-Algorithmus default

Ergebnisobjekt
list (breaks, counts, density)

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Box-Whisker-Diagramm

`boxplot (x, ..., notch, out, names, log, horizontal, add, ...)`

US air pollution

Übersichtsstatistik

Taille	Median
Kasten	($\frac{1}{4}, \frac{3}{4}$)-Quantile
Strichel	Wertebereich*
Kringel	Ausreißer
Kerbe	Signifikanz

Datenargumente

ein Vektor	x
mehrere Vektoren	x, ...
eine Matrix	x
ein Datensatz	x
Modellformel	x, data

```
require (HSAUR2)
boxplot (USairpollution, log="y", notch=TRUE,
         col=cm.colors (length (USairpollution)), main="US air pollution")
```

?character ?data.frame E/A Pakete Grafik 1D/2D-Plots 3D/HD-Plots lattice

Balkendiagramm für Vektor- und Matrixdaten

`barplot (height, width=1, legend.text, beside, horiz, col, axes, ...)`

Rural Male **Rural Female** **Urban Male** **Urban Female**

50-54 **55-59** **60-64** **65-69** **70-74**

Eingabedaten
Zahlenmatrix/vektor
benannte Zeilen
benannte Zeilen
positive Werte (s.u.)
moderate Anzahl

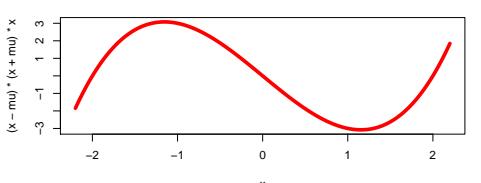
Balkengrafik
Hochstapler `beside=FALSE`
Orgelpfeifen `beside=TRUE`

VADeaths			
RM	RF	UM	UF
11.7	8.7	15.4	8.4
18.1	11.7	24.3	13.6
26.9	20.3	37.0	19.3
41.0	30.9	54.6	35.1
66.0	54.3	71.1	50.0

```
for(tb in list (VADeaths, t(VADeaths)))
  for (bs in c(FALSE,TRUE))
    barplot (height=tb, legend=TRUE, beside=bs, args.legend=list(x="top"))
```

Kurvenverlauf (Ausdrucksobjekt)

curve (expr, from, to, n=101, add=FALSE, type='l', ylab, log, ...)

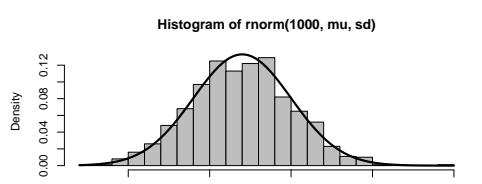


Ausdruck in 'x'

Analytisch \rightsquigarrow Stützwerte:
 $x <- seq(from,to,len=n)$
 $y <- eval(expr)$

Sekundärfunktion

Setze add=TRUE
(Koordinatenübernahme)

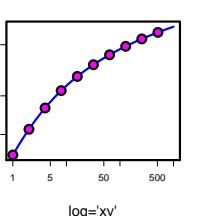
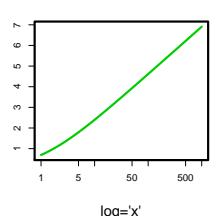
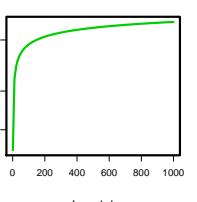
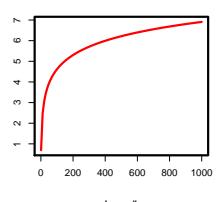


```
mu=2; sd=3;
curve ((x-mu)*(x+mu)*x, from=-1.1*mu, to=+1.1*mu, col="red", lwd=5)
```

```
hist (rnorm (1000,mu,sd), freq=FALSE, col="gray", breaks=32, xlab="")
curve (dnorm (x,mu,sd), add=TRUE, lwd=3)
```

Kurvenverlauf (Funktionsobjekt)

curve (x, y=0, to=1, from=y, xlim=NULL, ...) oder gleichwertig plot (< dto.>)



Funktionsobjekt

mit einem Argument:
 $log1plus <- function(a)$
 $log(1+a)$

Logarithm. Achsen

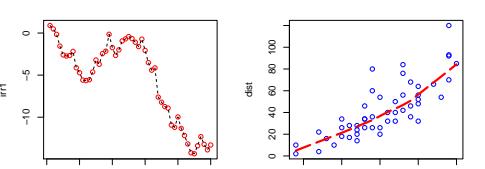
Argument $log = \begin{cases} \text{'x'} \\ \text{'y'} \\ \text{'xy'} \end{cases}$

Skalierungsweise ist transparent für sukzessives Zeichnen!

```
for (ll in c("", "x", "y", "xy"))
  curve (log1plus, 1, 1000, log=ll, col=2+nchar(ll), xlab="",
         sub=paste ("log=", ll, "", sep=""))
z <- 2^(0:9);   points (z, log1plus(z), pch=21, cex=2, bg="magenta")
```

Punkte & Linien in der Koordinatenebene

points (x, y=NULL, type='p', pch, cex, bg, ...) oder
lines (x, y=NULL, type='l', lwd, lty, ...)

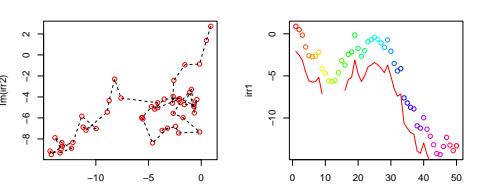


Punkte

Attributwechsel Punkt zu Punkt

Linien

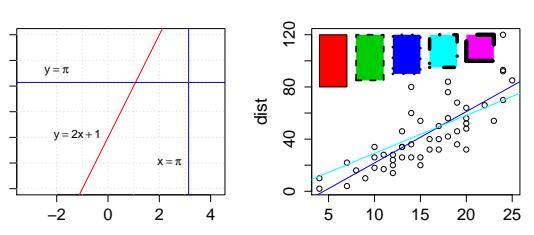
Unterbrechungen via NA
kein Attributwechsel



```
x <- rnorm(50); y <- rnorm(50); irr1 <- cumsum (x); irr2 <- cumsum (x + i*i*y)
plot (irr1, col="red"); lines (irr1, lty=2)
plot (irr2, col="red"); lines (irr2, lty=2)
plot (cars, col="blue"); lines (lowess (cars), col="red", lwd=3, lty="82")
plot (irr1, col=rainbow(50)); irr1[10:15] <- NA; lines (irr1-3, col=rainbow(50))
```

Geraden und Rechtecke

abline (a, b, h, v, reg, coef, ...) oder
rect (xleft, ybottom, xright, ytop, col, border, lty, lwd, ...)



```
plot (c(-2,3), c(-1,5), type="n", xlab="", ylab="", asp=1)
abline (h=-1:5, v=-2:3, col="lightgray", lty=3)
abline (a=1, b=2, col="red")
abline (h=pi, v=pi, col="blue")
```

```
attach (cars)
plot (cars)
abline (lsfit (speed, dist), col="blue")
abline (lsfit (speed, dist, intercept=FALSE), col="cyan")

N <- 1:5
rect (xleft=0+4*N, ybottom=75+5*N, xright=3+4*N, ytop=120,
      col=1:N, lty=N, lwd=N)
```

Geradenspezifikation

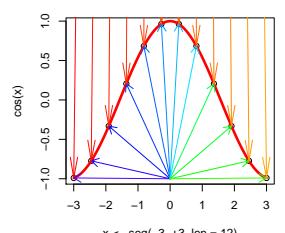
Koeffizienten a und b
Horizontale h
Vertikale v
Linearmodell reg
Koeffiz.vektor coef
(≥ 1 Geraden ok für h/v)

Boxenspezifikation

links unten xleft ybottom
rechts oben xright ytop
Füllfarbe(n) col
Randfarbe(n) border
Randattribut(e) lty lwd

Pfeile und Polygone

arrows (x0, y0, x1=x0, y1=y0, length, angle, code, ...) oder
polygon (x, y, density, angle, border, col, lty, ...)



```
plot (x <- seq (-3, +3, len=12), cos(x))
curve (cos, add=TRUE, col="red", lwd=3)
arrows (0, -1, x, cos(x), length=0.1, col=topo.colors(24))
arrows (0, +100, x, cos(x), angle=10, col=heat.colors(24))

x <- seq (0, 2012, len=(n<-12))
y <- cumsum (rnorm (n))
s <- abs (rnorm (n,3))
plot (c(x,x), c(y-s,y+s))
polygon (c(x,rev(x)), c(y-s,rev(y+s)), col="yellow", border="red")
lines (x, y, lty=3, col="blue")
```

Pfeilparameter

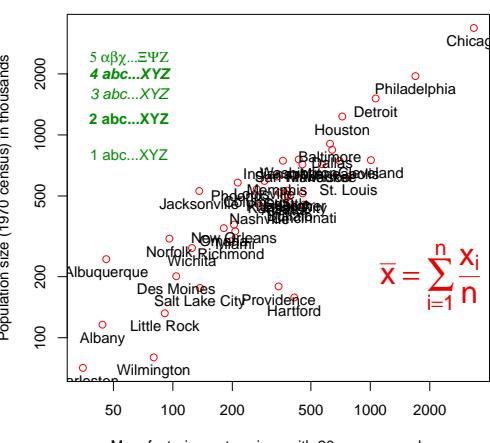
Startposition	x0 y0	
Zielposition	x1 y1	
Pfeilgeometrie	length angle	
Pfeiltyp	$\left\{ \begin{array}{l} \leftarrow \\ \rightarrow \\ \leftrightarrow \end{array} \right\}$	code $\left\{ \begin{array}{l} 1 \\ 2 \\ 3 \end{array} \right\}$

Polygonparameter

Ankerpunkte	x y
Schraffur	density angle
Füllfarbe	col
Randfarbe	border
Randattribute	lty lwd
Füllstrategie	fillOddEven

Textmarkierungen in der Koordinatenebene

text (x, y, labels, adj, pos, offset, vfont, cex, col, font, ...)



Textposition
pos, adj, und offset

Schriftattribute

cex, col und font
 $\left\{ \begin{array}{l} 1 = \text{normal} \\ 2 = \text{fettig} \\ 3 = \text{kursiv} \\ 4 = \text{beides} \\ 5 = \text{symbol} \end{array} \right\}$

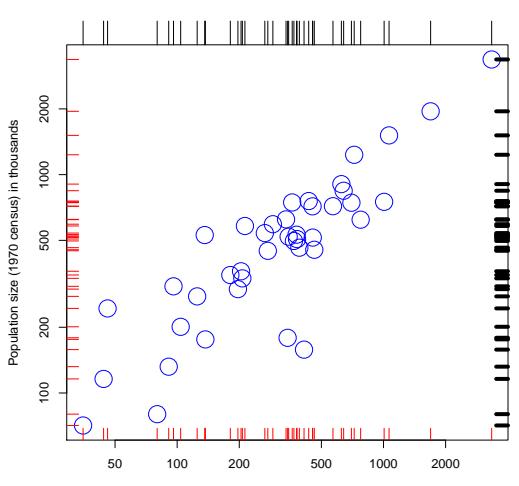
Texte abkürzen!

abbreviate (names, min=4, strict=FALSE)

```
plot (popul ~ manu, data=USairpollution, xlab=mlab, ylab=plab, log="xy", col="red")
text (popul ~ manu, data=USairpollution, pos=1, labels=rownames (USairpollution))
text (rep(60,5), 400*2:6, labels=paste (1:5, "abc...XYZ"), font=1:5, col="green4")
text (2000, 200, cex=2, col="red2", labels=expression (bar(x) == sum(frac(x[i], n), i==1, n)))
```

Stützstellenmarkierungen an Koordinatenachse

rug (x, ticksize=0.03, side=1, lwd=0.5, col=par('fg'), quiet=, ...)



Achsenwahl

1=unten 2=links 3=oben
4=rechts

Koordinaten

Übernahme, auch log/log

Markierungsgrößen

Paperskala, nicht Datenskala

Doubletten schütteln!

jitter (x,
factor=1, amount=NULL)

Zeichenketten — spezielle Funktionen

Datensätze — spezielle Funktionen

Eingabefunktionen & Ausgabefunktionen

'R'-Programmpakete — installieren & laden

Grundlegende Mechanismen der 'R'-Grafikfunktionen

Univariate und bivariate Darstellungen

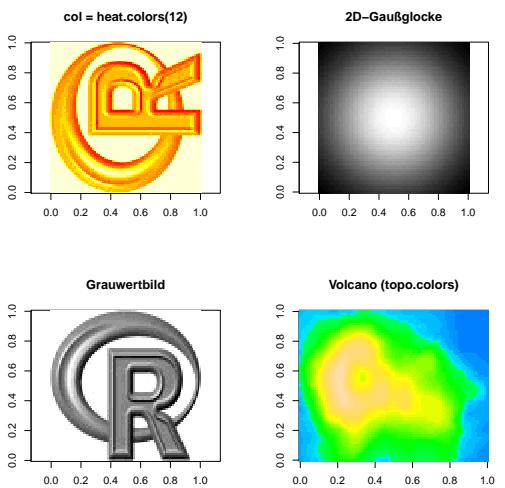
Räumliche und vieldimensionale Darstellungen

Das Trellis-Grafiksubsystem

```
plot (popul ~ manu, data=USairpollution, xlab=mlab, ylab=plab, log="xy", col="blue", cex=3)
rug (USairpollution$manu, side=1, col="red"); rug (USairpollution$manu, side=3, ticks=-0.06)
rug (USairpollution$popul, side=2, col="red"); rug (USairpollution$popul, side=4, lwd=5)
```

Rastergrafik für $f(x, y)$

image (x, y, z, zlim, xlim, ylim, col=heat.colors(12), add, breaks, ...)



Geometrie

Definitionsbereich
Wertematrix
Pixel quadratisch
Gebietsgrenzen $xlim\ ylim\ zlim$
neue Koordinaten? $add=FALSE$

x y
z
asp=1

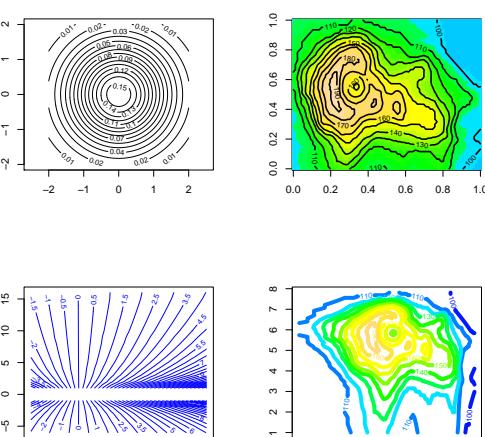
Färbung

Farbpalette
Farbgrenzen
col
breaks

```
library (jpeg); logo <- readJPEG("Rlogo.jpg")[,1]
image (logo, asp=1, main="col = heat.colors(12)")
image (t(logo[nrow(logo):1,]), asp=1, col=gray (0:20/20), main="Grauwertbild")
x <- dnorm (-30:+30/20)
image (z=x%/%x, col=gray (0:50/50), asp=1, main="2D-Gaußglocke")
image (volcano, col=topo.colors(48)[-1:10], main="Volcano (topo.colors)")
```

Konturgrafik für $f(x, y)$

contour (x, y, z, nlevels=10, levels, labels, method, col, lty, add, ...)



Geometrie

Definitionsbereich
Wertematrix
Anzahl Schichten
Konturhöhenvektor
Gebietsgrenzen $xlim\ ylim\ zlim$
neue Koordinaten? $add=FALSE$

x y
z
nlevels
levels

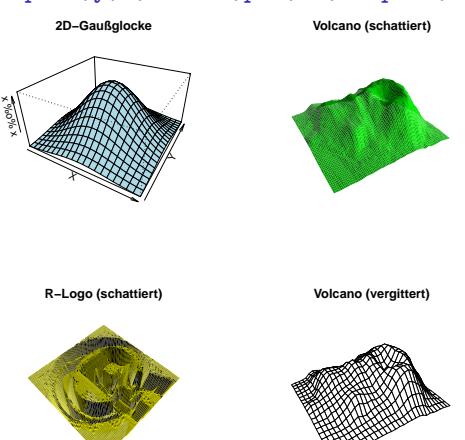
Beschriftung

Konturmarkierungen $labels$
Schriftattribute $vfont\ labcex$
Linienattribute $col\ lty\ lwd$

```
x <- seq (-2, +2, len=100); contour (x=x, y=y, z=outer (dnorm(x), dnorm(y)), nlevels=12, asp=1)
x <- -6:16; z <- outer(x, sqrt(abs(x)), FUN="/"); contour(x,y,z, col="blue", method="edge", nlev=32)
image (z=volcano, col=topo.colors(32)[-1:10]); contour (z=volcano, add=TRUE, lwd=2)
contour (x=sqrt(1:nrow(volcano)), y=sqrt(1:ncol(volcano)), z=volcano, col=topo.colors(12), lwd=5)
```

Perspektivgrafik für $f(x, y)$

persp (x,y,z, theta,phi, r, expand, col, border, shade, box, ...)



Geometrie

Definitionsbereich
Wertematrix
Azimuthwinkel
Colatitudewinkel
Auge-Box-Distanz
Aspekttreue $scale=FALSE$
z-Expansion $expand$
Gebietsgrenzen $xlim\ ylim\ zlim$

x y
z
theta
phi
r

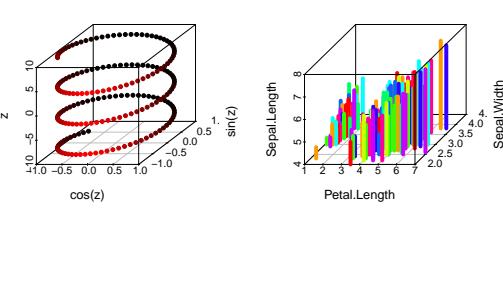
Licht und Farbe

Facettenfarbe(n) col
Gitterfarbe(n) $border$
Schattierungskoeff. $shade$
Lichtquelle $ltheta\ lphi$
3D-Beachung? box

```
x <- dnorm (-10:+10/5)
persp (z=x%/%x, theta=30, phi=30, expand=0.5, col="lightblue", main=
persp (z=sqrt(logo), theta=45, phi=65, shade=0.75, col="yellow", border=NA, box=FALSE, main=
persp (z=volcano, theta=150, phi=30, expand=0.4, col="green", shade=0.75, box=F, border=NA, main=
persp (z=volcano [1:nrow(volcano)%3==0, 1:ncol(volcano)%3==0,
theta=150, phi=30, expand=0.4, box=FALSE, main=
```

3D Scatterplot für (x, y, z) -Punktwolken

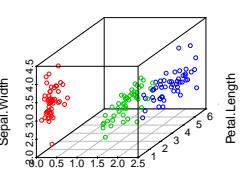
scatterplot3d (x,y,z, color, highlight.3d, x/y/zlim, x/y/zlab, scale, angle, grid, box, type, col.*., cex.*., font.*., lty.*., ...)



Geometrie

Punktwolke
 (x, y) -Winkel
y-Achsenfaktor $scale.y$
Gebietsgrenzen $xlim\ ylim\ zlim$

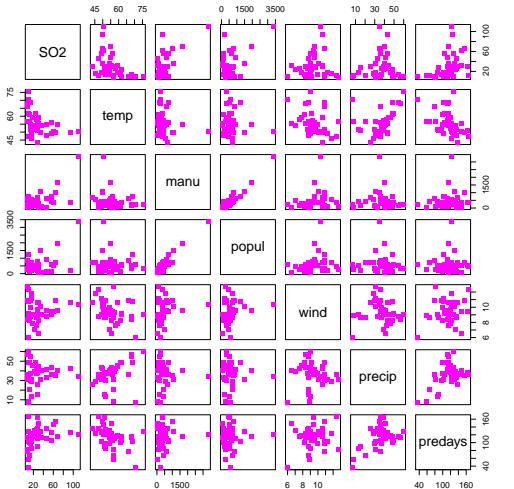
x y z
angle
scale.y



```
z <- seq (-10, +10, 0.1); scatterplot3d (x=cos(z), y=sin(z), z=z, pch=19, highlight.3d=TRUE)
scatterplot3d (iris[4:2], color=1+unclass(iris$Species))
scatterplot3d (iris[3:1], type="h", pch=" ", lwd=5, color=rep(rainbow(10),15))
rb <- rainbow (15); rb3 <- t (col2rgb (rb)); scatterplot3d (rb3, color=rb, box=FALSE, axis=F) -> geo
tp <- cm.colors (15); tp3 <- t (col2rgb (tp)); geo$points3d (tp3, pch="X", col=tp)
```

Scatterplot: $n \times n$ 2D-Wolken

```
pairs (x, labels, panel, ..., cex.labels, font.labels, gap=1, ...)
```



pairs-Parameter

Matrix/Datensatz **x**
Variablenamen **labels**
Zwischenraum **gap**
points-Argumente **...**

USairpollution

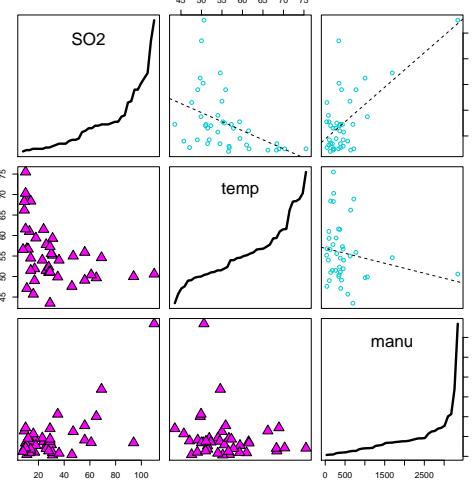
Luftverschmutzung in 41 US-Metropolen

SO_2 -Gehalt in $\mu g/m^3$
 \varnothing -Temperatur in deg F
Firmen (≥ 20 Mitarb.)
Einwohner (Tsd., 1970)
Windgeschwindigkeit mph
Niederschlag inches
Regentage 1/J

```
library (HSAUR2)
pairs (USairpollution, cex=8, pch=". ", col="magenta")
```

Scatterplot mit individuellem Layout

```
pairs (x, labels, panel, lower.panel, upper.panel, diag.panel,
text.panel, ...)
```

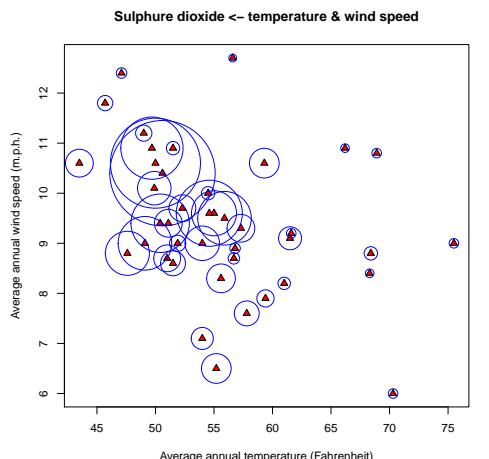


library (HSAUR2)

```
pairs (USairpollution[c(1,2,3)],
lower.panel=points,
upper.panel=function (x,y,...)
{
  points (x, y, col="cyan3")
  abline (lsfit(x,y), lty=2)
},
diag.panel=function (x,...)
  lines (
    seq (
      min(x),
      max(x),
      len=length(x)
    ),
    sort(x), lwd=3, ...
  ),
cex=2, pch=24, bg="magenta")
```

Ikonische 3D-Darstellung: „bubble plot“

```
symbols (x, y, circles, squares, inches, add=FALSE, fg, bg, ...)
```



Dritte Dimension

als Symbolgröße kodiert
($z \geq 0$)

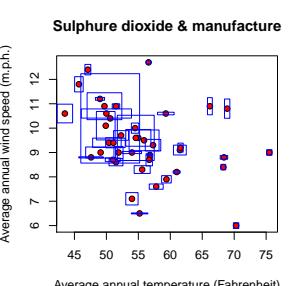
Funktionsargumente

Punktepositionen **x y**
z-Koordinate **circles**
oder **squares**
Maximalgröße **inches**
(automatisch) **inches=FALSE**
Berandungsfarbe **fg**
Hintergrundfarbe **fg**

```
with (USairpollution,
plot (temp, wind, main=slab, xlab=tlab, ylab=wlab, pch=24, bg="red"))
with (USairpollution,
symbols (temp, wind, add=TRUE, circles=S02, inches=0.75, fg="blue"))
```

Ikonische Darstellung von $\geq 3D$ -Punktwolken

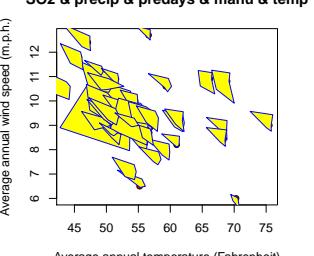
```
symbols (x, y, rectangles, stars, thermometers, boxplots, ...)
```



Zusatzdimensionen
als Symbolattribute kodiert
($z_i \geq 0$)

Geometrie

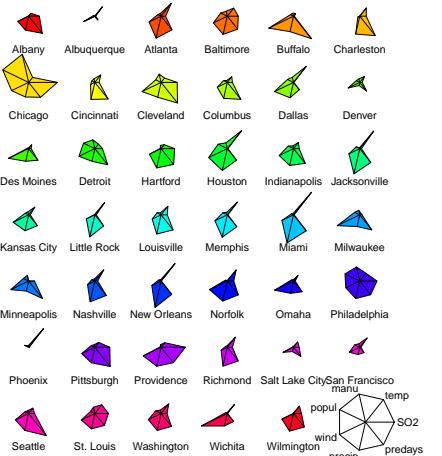
Punktepositionen **x y**
 z_1, \dots, z_2 **rectangles**
 z_1, \dots, z_3 **thermometers**
 z_1, \dots, z_4 **thermometers**
 z_1, \dots, z_5 **boxplots**
 z_1, \dots, z_3, \dots **stars**



Alle z_i -Argumente als Matrizen
oder Datensätze passender
Dimension

Sterngrafik

```
stars (x, full, scale, radius, labels, location, nrow, ncol, len,
key.loc, draw.segments=FALSE, col.segments, col.stars, ...)
```



Funktionsargumente

Matrix, Datensatz	x
Voll/halbkreis?	full
Radiallinien?	radius
Objektnamen	labels
Legendenposition	key.loc
Attributnamen	key.labels
Objektfarben	col.stars
Skalenfaktor	len

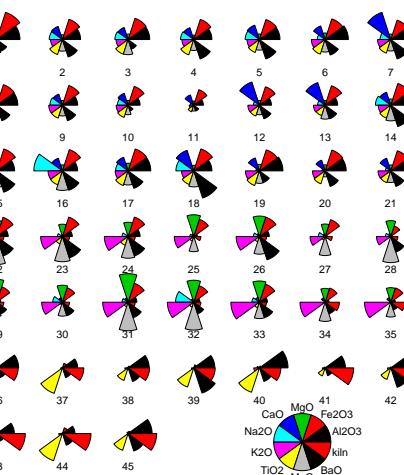
Objektgeometrie

2D-Positionen	locations
Gitterdimensionen	nrow ncol

```
stars (USairpollution, draw.segments=FALSE,
col.stars=rainbow(41), key.loc=c(14,2), flip.labels=FALSE, cex=0.8)
```

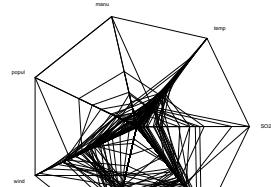
Radargrafik und Spinnennetz

```
stars (x, full, scale, radius, labels, location, nrow, ncol, len,
key.loc, draw.segments=TRUE, col.segments, col.stars, ...)
```



Funktionsargumente

Matrix, Datensatz	x
Attributfarben	col.segments
Spinnennetz	loc=c(0,0)

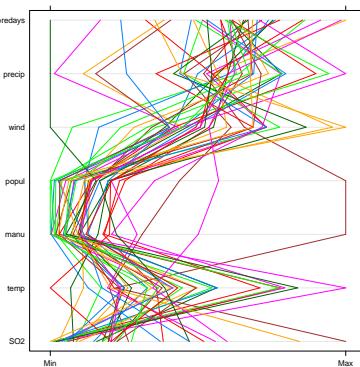


```
stars (USairpollution,
draw.segments=FALSE,
locations=c(0,0), key.loc=c(0,0))
```

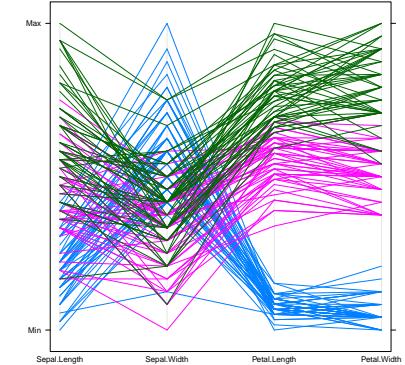
```
require (HSAUR2); stars (pottery, draw.segments=TRUE, key.loc=c(13,2))
```

Wäschleinengrafik: parallele Koordinaten

```
parallel (x, data=NULL, groups=NULL, subset=TRUE, ...)
```

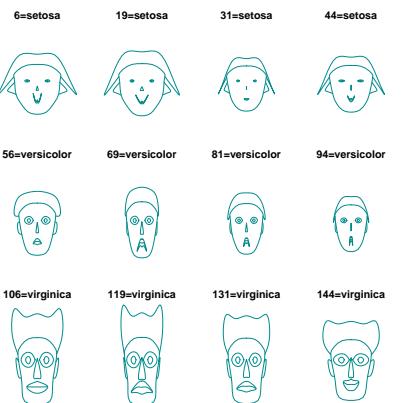


```
require (lattice)
parallel (x=iris[1:4],
groups=iris$Species,
horizontal.axis=FALSE
)
```



Chernoff-Gesichter: physiognomische Darstellung

```
faces (xy, fill=FALSE, nrow, ncol, scale=TRUE, labels)
```



Chernoff-Merkmale

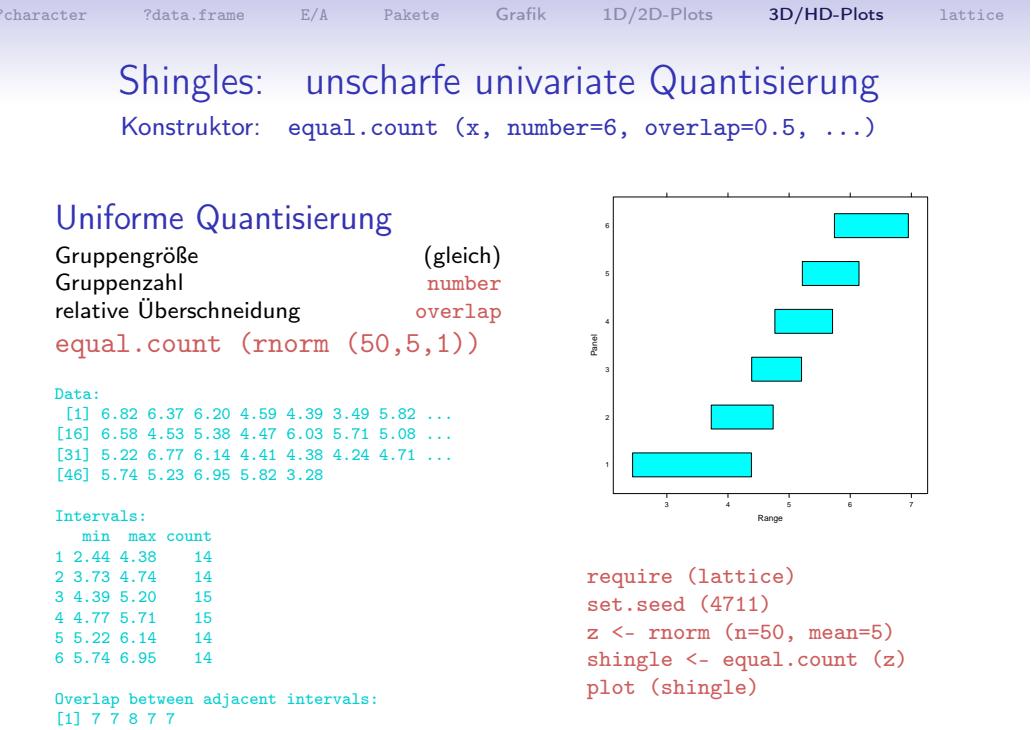
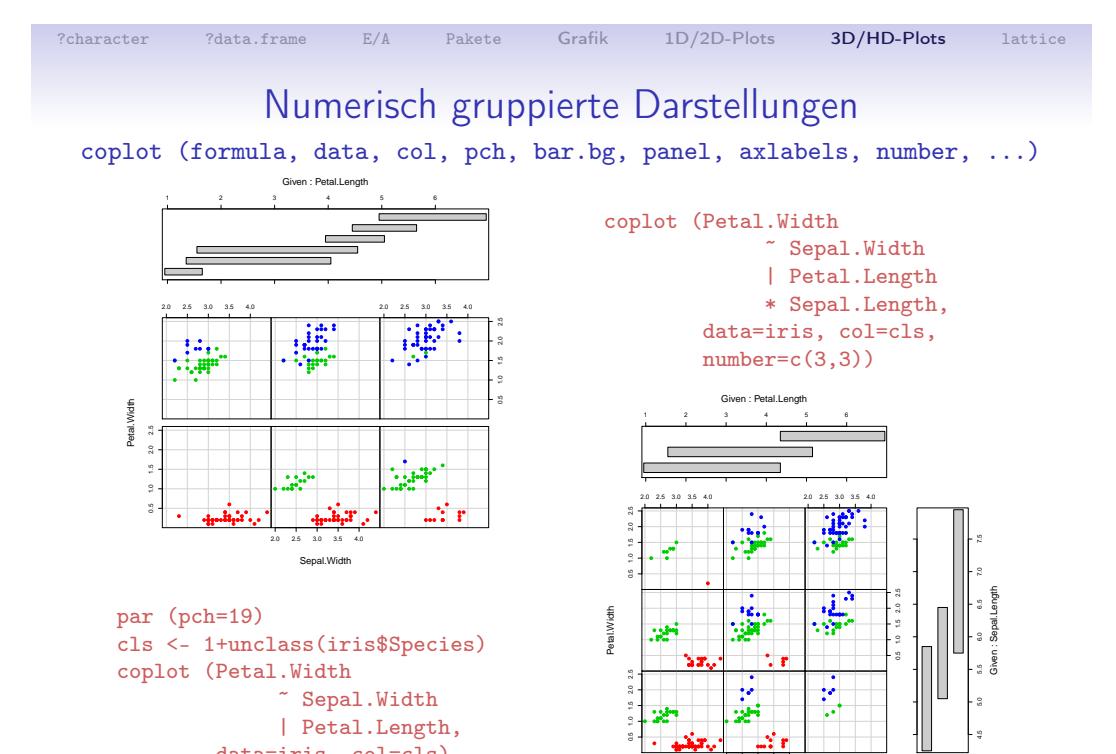
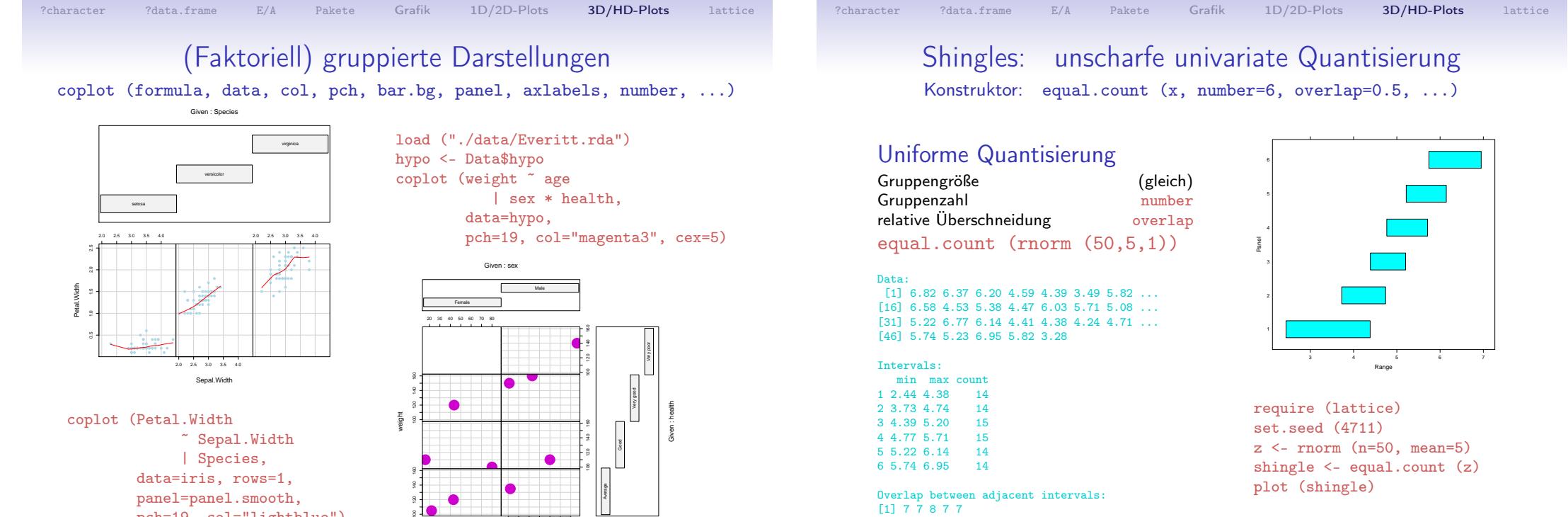
1-height of face · 2-width of face ·
3-shape of face · 4-height of mouth ·
5-width of mouth · 6-curve of
smile · 7-height of eyes · 8-width of
eyes · 9-height of hair · 10-width of
hair · 11-styling of hair · 12-height
of nose · 13-width of nose ·
14-width of ears · 15-height of ears.

Funktionsargumente

Numerische Datenmatrix	xy
Gitterlayout	ncol nrow
Attribute normieren?	scale
Attribute ~ PCA?	fill

```
require (TeachingDemos)
N=150; M=4; K=3; idx <- round ((1:(K*M)-.5)*N/K/M)
lab <- paste (idx, iris$Species[idx], sep="=")
faces (iris[idx,-5], nrow=K, ncol=M, labels=lab, fill=FALSE)
```

```
require (lattice)
require (HSAUR2)
parallel (x=USairpollution)
```



- Zeichenketten — spezielle Funktionen
- Datensätze — spezielle Funktionen
- Eingabefunktionen & Ausgabefunktionen
- 'R'-Programmpakete — installieren & laden
- Grundlegende Mechanismen der 'R'-Grafikfunktionen
- Univariante und bivariate Darstellungen
- Räumliche und vieldimensionale Darstellungen
- Das Trellis-Grafiksubsystem

Zusammenfassung (3)

1. Zeichenketten sind **atomare** 'R'-Typen; es gibt also keine expliziten Einzelzeichen.
2. **Datensätze** werden durch die Klasse `data.frame` repräsentiert.
Zeilen enthalten die **Objekte** (Fälle) und Spalten die numerischen, nominalen oder ordinalen **Attribute** (Variable, Merkmale).
3. Die **E/A** geschieht mit `save()` und `load()` (**komplexe** Typen) oder mit `write()` und `scan()` (**atomare** Typen).
4. **Grafiken** werden **sukzessiv** erzeugt; ein Primäraufruf baut (u.a.) das **Koordinatensystem**, nachfolgende Aufrufe vervollständigen die Grafik mit Punkten, Linien, Flächen, Texten und Ikonen.
5. Mehrere Grafiken können auf einer **Leinwand** arrangiert werden.
6. Wir unterscheiden Darstellungstechniken für **Verteilungseigenschaften** (z.B. Histogramme), für **Punktwolken** (Scatterplots) und für **Funktionsverläufe** (Linien, Flächen).
7. Die Visualisierung **Höherdimensionale** Daten ($dim \geq 4$) erfolgt durch **Ikonenbildung** (Sterne, Gesichter), **Staffelung** (Wäscheleinen) oder **Schachtelung** (`pairs()`, `coplot()`).

WERKZEUGE DER MUSTERERKENNUNG UND DES MASCHINELLEN LERNENS

Teil IV

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 19. Februar 2018

Modellieren in 

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

-Funktionen für Wahrscheinlichkeitsverteilungen

'R' bietet systematisch **vier** Funktionen je Verteilungsfunktion an

Verteilungsdichtefunktion

dname (x, ...)

Berechnet die Dichtewerte $f(x_i)$ an den Stellen x[i].

Verteilungsparameter werden als (benannte) Argumente übergeben.

Kumulative Verteilungsfunktion

pname (q, ...)

Berechnet Wahrscheinlichkeiten $P(\mathbb{X} \leq q_i)$ an den Stellen q[i].

Quantile

qname (p, ...)

Berechnet die Werte $q_i \in \mathbb{R}$ mit $P(\mathbb{X} \leq q_i) = p[i]$.

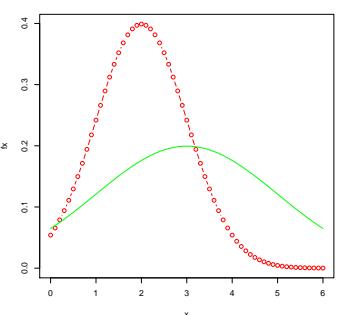
Zufallswerte

rname (n, ...)

Würfelt n $\in \mathbb{N}$ viele Zufallswerte unter der spezifizierten Verteilung aus.

Dichtewerte einer Verteilung

```
dname(x, «param1» ..... «paramK», log=FALSE)
```



```
x <- seq (0,6,by=0.1)
fx <- dnorm (x, mean=2, sd=1, log=F)
plot (x, fx, col="red", type="b")

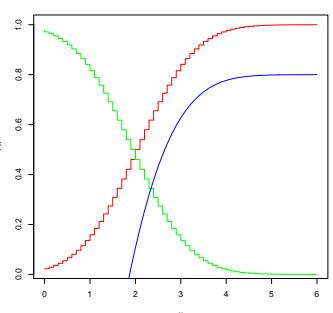
gx <- dnorm (x, mean=3, sd=2, log=F)
lines (x, gx, col="green", type="l")
```

Bemerkung

Die Einstellung `log=TRUE` bewirkt die Logarithmierung der Dichtewerte zur Basis e.

Kumulative Verteilungswahrscheinlichkeiten einer Dichte

```
pname(q, «param1» ..... «paramK», lower.tail=TRUE, log.p=FALSE)
```



```
x <- seq (0,6,by=0.1)
Fx <- pnorm (q=x, mean=2, sd=1)
plot (x, Fx, col="red", type="s")

Gx <- pnorm (q=x, mean=2, sd=1, low=F)
lines (x, Gx, col="green", type="S")

logFx <- pnorm (q=x, mean=2, sd=1, log=T)
lines (x, 0.8+logFx, col="blue", type="l")
```

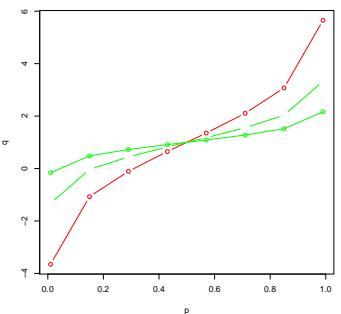
Bemerkung

Die Einstellung `log.p=TRUE` bewirkt die Logarithmierung der Wahrscheinlichkeitswerte zur Basis e.

Mit `lower.tail=FALSE` werden die Werte $P(\bar{X} > q_i)$ statt $P(\bar{X} \leq q_i)$ berechnet.

Quantilwerte einer Verteilung

```
qname(p, «param1» ..... «paramK», lower.tail=TRUE, log.p=FALSE)
```



```
p <- seq (0.01,0.99,length=8)
q <- qnorm (p=p, mean=1, sd=2)
plot (p, q, col="red", type="b")

q <- qnorm (p=p, mean=1, sd=1)
lines (p, q, col="green", type="c")

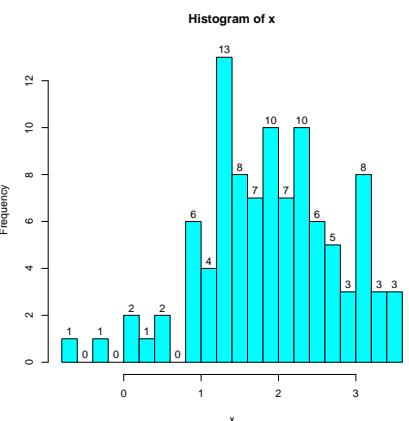
q <- qnorm (p=p, mean=1, sd=0.5)
lines (p, q, col="green", type="o")
```

Bemerkung

Die Einstellung `log.p=TRUE` bewirkt die Interpretation der vorgegebenen Wahrscheinlichkeitswerte als Logarithmen zur Basis e.

1D Sampling — Auswürfeln von Zufallswerten

```
rname(n, «param1» ..... «paramK»)
```



```
x <- rnorm (100, mean=2, sd=1)

hist (x, breaks=20,
      col="cyan",
      labels=T, plot=T)
```

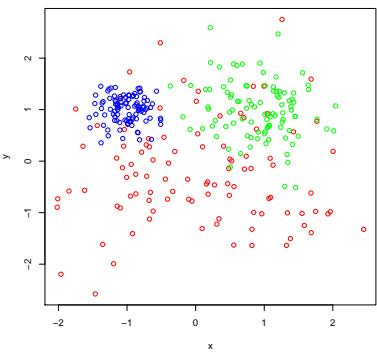
Bemerkung

Mit dem Kommando `set.seed (seed, kind)` kann der **Zufallszahlengenerator** in einen definierten Zustand $seed \in \mathbb{N}$ versetzt werden.

$$\text{kind} \in \left\{ \begin{array}{l} \text{'Wichmann-Hill'} \\ \text{'Marsaglia-Multicarry'} \\ \text{'Super-Duper'} \\ \text{'Mersenne-Twister'} \\ \text{'Knuth-TAOCP-2002'} \end{array} \right\}$$

2D Sampling — Auswürfeln von Zufallswerten

```
random(n, «param1» ..... «paramK»)
```



```
x <- rnorm (100, mean=0, sd=1)
y <- rnorm (100, mean=0, sd=1)
plot (x, y, col="red", type="p")

x <- rnorm (100, mean=1, sd=0.5)
y <- rnorm (100, mean=1, sd=0.5)
lines (x, y, col="green", type="p")

x <- rnorm (100, mean=-1, sd=0.25)
y <- rnorm (100, mean=1, sd=0.25)
lines (x, y, col="blue", type="p")
```

Sampling — Auswürfeln in endlichen W-Räumen

```
sample (x, size, replace=FALSE, prob=NULL)
```

- Ziehen ohne Zurücklegen

```
sample (x=0:9, size=5)           3 7 1 5 8
sample (x=letters, size=5)      'm' 'y' 'o' 'f' 'b'
sample (x=49, size=6)           14 13 30 23 25 45
```

- Zufallspermutation

```
sample (x=0:9)                  3 1 4 5 2 9 6 0 7 8
sample (x=6)                     1 6 4 2 5 3
```

- Ziehen mit Zurücklegen

```
sample (x=c(T,F), size=5, rep=T) TRUE FALSE FALSE TRUE FALSE
sample (x=c('C','G','A','T'), size=18, rep=TRUE)
'C' 'T' 'T' 'G' 'C' 'G' 'A' 'G' 'T' 'C' 'G' 'C' 'C' 'T' 'T' 'A' 'T' 'G' 'C'
```

- Vorgabe der Proportionen

```
sample (x=2, size=9, rep=T, prob=c(3,1)) 1 1 2 1 1 1 1 2 1
```

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf [0, 1]

Diskrete (univariate) Verteilungen

Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Arithmetische und getrimmte Mittelwerte

```
mean(x, trim=0, na.rm=FALSE, ...)
```

- Arithmetisches Mittel einer Zahlmenge

```
mean (1:100)                      [1] 50.5
mean (rnorm (1000, mean=17))       [1] 17.01375
```

- Mittelwerte der Attribute eines Datensatzes
data (iris); mean (iris)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.843333	3.057333	3.758000	1.199333	NA

- Mittelwert einer Zahlenmatrix

```
mean (as.matrix (iris[1:4]))        [1] 3.4645
```

- Getrimmte Mittelwerte (*Trimmt = Anteil der gelöschten unteren/oberen Ausreißer*)

```
mean (c(1,1,1,4,8), trim=0)         [1] 3
mean (c(1,1,1,4,8), trim=0.2)       [1] 2
mean (c(1,1,1,4,8), trim=0.4)       [1] 1
median (c(1,1,1,4,8))              [1] 1
```

Varianz und Standardabweichung

```
var (x, y=NULL, na.rm=FALSE, use)
```

- Varianz eines Datenvektors

```
var (1:5) [1] 2.5
var (rnorm (10**3, mean=7, sd=2)) [1] 4.230118
var (rnorm (10**6, mean=7, sd=2)) [1] 3.997247
```

- Standardabweichung eines Datenvektors

```
sd (rnorm (10**6, mean=7, sd=2)) [1] 2.000575
```

- Kovarianz zweier Datenvektoren

```
x <- rnorm (10**3)
var (x)
var (x, x)
var (x, x+17)
var (x, x+x)
y <- rnorm (10**3)
var (x, y)
[1] 0.9386478
[1] 0.9386478
[1] 0.9386478
[1] 1.877296
[1] 0.05117009
```

- (Ko)varianzberechnung nach Pearson

```
sum((x-mean(x))*(y-mean(y)))/(1000-1) [1] 0.05117009
```

Sortierung, Rang und Reihenfolge

- Sortieren von Werten eines Vektors

```
x <- sample (101:109)
sort (x)
sort (x, decreasing=TRUE)
rev (sort (x))
108 104 105 107 103 106 102 101 109
101 102 103 104 105 106 107 108 109
109 108 107 106 105 104 103 102 101
109 108 107 106 105 104 103 102 101
```

- Rangnummern ausrechnen

```
rank (x)
8 4 5 7 3 6 2 1 9
```

- Rangnummern ausrechnen — incl. Doubletten

```
x <- sample (1:9, replace=T)
rank (x, ties='average') 7.0 5.0 8.5 1.0 3.5 6.0 8.5 3.5 2.0
rank (x, ties='first') 7 5 8 1 3 6 9 4 2
rank (x, ties='random') 7 5 8 1 4 6 9 3 2
```

- Sortierindex erstellen

```
print (x)
order (x)
x [order (x)]
6 4 9 1 3 5 9 3 2
4 9 5 8 2 6 1 3 7
1 2 3 3 4 5 6 9 9
```

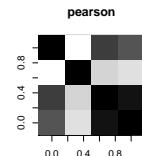
Kovarianz und Korrelation

```
cov (x, y=NULL, method = c('pearson', 'kendall', 'spearman'))
```

- Kovarianzen für die Spalten einer Matrix

```
cov (iris[1:4])
```

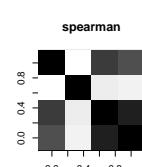
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.68569351	-0.04243400	1.2743154	0.5162707
-0.04243400	0.18997942	-0.3296564	-0.1216394
1.27431544	-0.32965638	3.1162779	1.2956094
0.51627069	-0.12163937	1.2956094	0.5810063



- Korrelationen für die Spalten einer Matrix

```
cor (iris[1:4], method='pearson')
```

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



Quantile — Rangordnungsstatistiken

- Wertebereichsgrenzen

```
set.seed (4711)
x <- sample (883:4711, size=101)
range (x)
884 4671
```

- Tukey-Synopse: 0%, 25%, 50%, 75%, 100%-Quantile

```
fivenum (x)
884 1892 3017 3828 4671
```

- Quantile (allgemein)

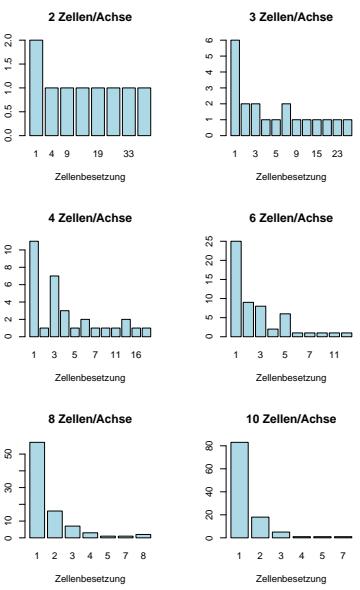
```
quantile (x, probs=0:2/2, type=7)
quantile(x,1/2) == median(x)
884 3017 4671
50% TRUE :-)
```

- Univariate Quantisierung

```
cut (19:11, breaks=3) -> f (3 Gruppen gleicher Größe)
levels (f) "(11,13.7]" "(13.7,16.3]" "(16.3,19]"
unclass (f)
3 3 3 2 2 2 1 1 1
cut (19:11, breaks=3, labels=LETTERS[1:3])
C C C B B B A A A
unclass (cut (19:11, breaks=c(10,15,20)))
2 2 2 2 1 1 1 1 1
```

Beispiel: der Fluch der Dimension

Äquidistante Vergitterung des Merkmalraumes (Hyperkuben)



```
for (m in c(2,3,4,6,8,10)) {
  A <- sapply (
    iris[1:4],
    cut, breaks=m,
    labels=LETTERS[1:m])
  S <- apply (A, MARGIN=1,
             paste, collapse="")
  f <- table (S)
  F <- table (f)
  barplot (F,
            main=paste (m, "Zellen/Achse"),
            xlab="Zellenbesetzung",
            col="lightblue")
}
```

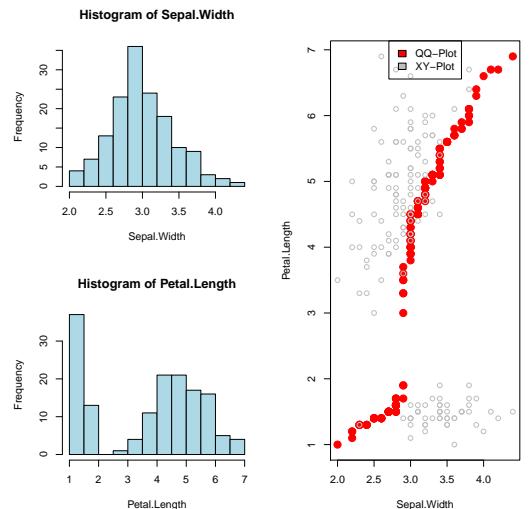
Verfahrensweise

achsenweise quantisieren
Zellencode bilden
Zellenbesetzung zählen
Größenstatistik berechnen

A
S
f
F

Beispiel: Verteilungsähnlichkeit vs. Korrelation

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



```
attach (iris)
layout (matrix (c(1,2,3,3), 2))

hist (Sepal.Width,
      col="lightblue")
hist (Petal.Length,
      col="lightblue")

qqplot (
  Sepal.Width, Petal.Length,
  col="red", pch=19, cex=1.5)
points (
  Sepal.Width, Petal.Length,
  col="gray")
legend ("top",
        legend=c("QQ-Plot", "XY-Plot"),
        fill=c("red", "gray"))
```

Vergleich zweier Verteilungen

Probability-Probability-Plot versus Quantile-Quantile-Plot

Univariate Verteilung

Verteilungsdichte

$$f : \mathbb{R} \rightarrow \mathbb{R}_0^+$$

Kumulative Verteil.funktion

$$F : \begin{cases} \mathbb{R} & \rightarrow [0, 1] \\ x & \mapsto F(x) := \int_{-\infty}^x f(\xi) d\xi \end{cases}$$

Quantilfunktion

$$Q : \begin{cases} [0, 1] & \rightarrow \mathbb{R} \\ p & \mapsto Q(p) := F^{-1}(p) \end{cases}$$

Verteilungsvergleich:

$(f_1, f_2), (F_1, F_2), (Q_1, Q_2)$?

PP-Darstellung

Graph der Punktmenge

$$\langle F_1(x), F_2(x) \rangle_{x \in \mathbb{R}}$$

QQ-Darstellung

Graph der Punktmenge

$$\langle Q_1(p), Q_2(p) \rangle_{p \in [0,1]}$$

Empirische Formulierung

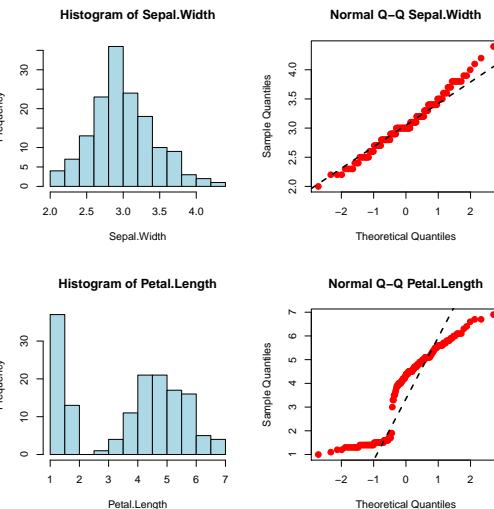
Datensätze $\{y_1, \dots, y_m \sim f_1\}$
 $\{z_1, \dots, z_m \sim f_2\}$

QQ-Plot $\langle y_{(j)}, z_{(j)} \rangle$

Scatterplot $\langle y_j, z_j \rangle$

Normalverteilungseigenschaft einer Punktmenge

`qqnorm (y, ylim, main=, xlab=, ylab=, plot.it=TRUE, datax=FALSE, ...)`



```
attach (iris)
layout (matrix (c(1,2,3,4), 2))

hist (Sepal.Width,
      col="lightblue")
hist (Petal.Length,
      col="lightblue")

qqnorm (Sepal.Width,
        main="Normal Q-Q Sepal.Width",
        col="red", pch=19, cex=1.2)
qqline (Sepal.Width, lty=2, lwd=2)

qqnorm (Petal.Length,
        main="Normal Q-Q Petal.Length",
        col="red", pch=19, cex=1.2)
qqline (Petal.Length, lty=2, lwd=2)
```

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

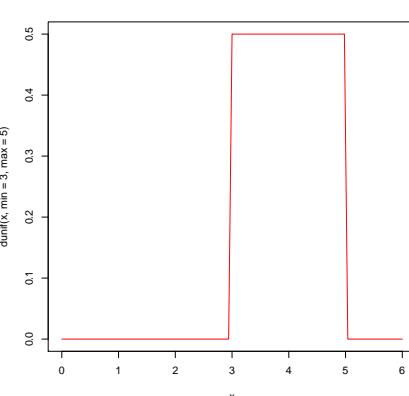
Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Gleich-, Rechteck- oder uniforme Verteilung

`dunif (x, min=0, max=1)`

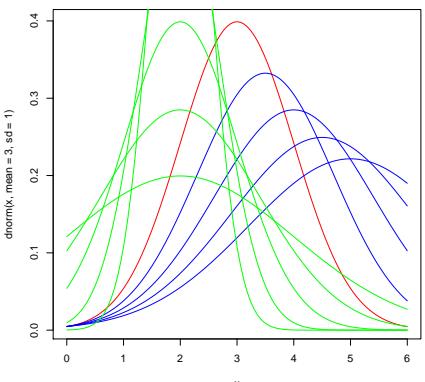


- $x \in \mathbb{R}$
- a Minimalwert
- b Maximalwert
- $E[\mathbb{X}] = \frac{(a+b)}{2}$
- $\text{Var}[\mathbb{X}] = \frac{(b-a)}{12}$

$$f_{a,b}(x) = \frac{1}{b-a}$$

Normalverteilung

`dnorm (x, mean=0, sd=1)`

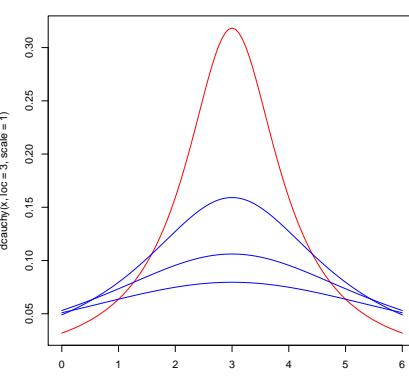


- $x \in \mathbb{R}$
- μ Erwartungswert
- $\sigma > 0$ Standardabweichung
- $E[\mathbb{X}] = \mu$
- $\text{Var}[\mathbb{X}] = \sigma^2$

$$f_{\mu,\sigma}(x) = \mathcal{N}(x | \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp \left\{ -\frac{1}{2} \cdot \left(\frac{x-\mu}{\sigma} \right)^2 \right\}$$

Cauchyverteilung

`dcauchy (x, location=0, scale=1)`



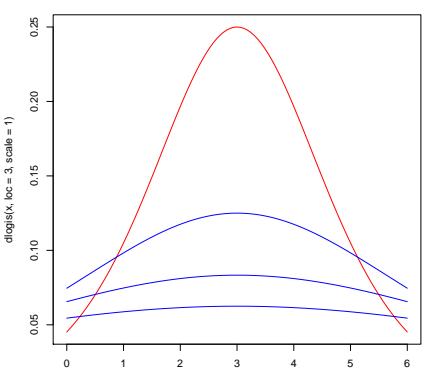
- $x \in \mathbb{R}$
- μ Lageparameter
- s Skalenparameter
- $E[\mathbb{X}] = \mu$
- $\text{Var}[\mathbb{X}] = \infty$

$$f_{\mu,s}(x) = \left\{ \pi s \cdot \left(1 + \left(\frac{x-\mu}{s} \right)^2 \right) \right\}^{-1}$$

„Schwarzeneggerdichte“ (schmaler Kopf, extrem breite Schultern)

Logistische Verteilung

`dlogis(x, location=0, scale=1)`



$x \in \mathbb{R}$

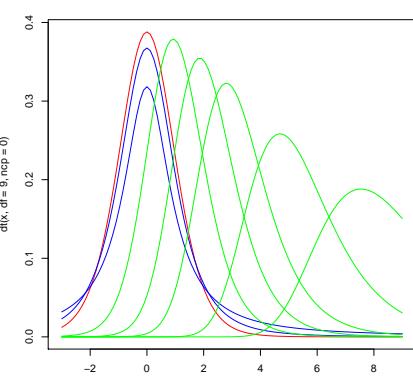
μ Lageparameter

s Skalenparameter

$$\mathcal{E}[\mathbb{X}] = \mu$$

$$\text{Var}[\mathbb{X}] = \frac{\pi^2}{3} \cdot s^2$$

$$f_{\mu,s}(x) = \frac{1}{s} \cdot e^{\frac{x-\mu}{s}} \cdot \left\{ 1 + e^{\frac{x-\mu}{s}} \right\}^{-2} \quad \text{bzw.} \quad F_{\mu,s}(x) = \left\{ 1 + e^{-\frac{x-\mu}{s}} \right\}^{-1}$$



$x \in \mathbb{R}$

n Anzahl Freiheitsgrade

ncp Nicht-Zentralität

$$\mathcal{E}[\mathbb{X}] = 0$$

$$\text{Var}[\mathbb{X}] = \frac{n}{(n-2)}$$

$$f_n(x) = \frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi} \cdot \Gamma(\frac{n}{2})} \cdot \left(1 + \frac{x^2}{n} \right)^{-\frac{n+1}{2}}$$

Die Variable $T = (\hat{\mu} - \mu_0) \cdot \sqrt{n}/\sigma$ von n i.i.d. $\mathcal{N}(\mu, \sigma^2)$ -verteilten ZVen ist Student ($n-1, t^*$)-verteilt mit $t^* = (\mu - \mu_0) \cdot \sqrt{n}/\sigma$

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

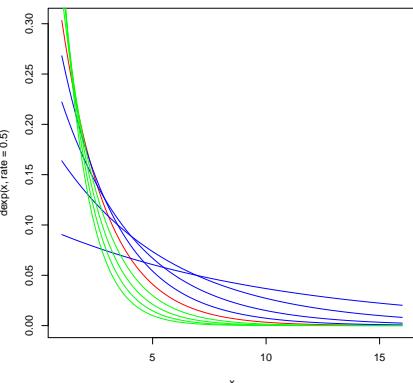
Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Exponentialverteilung

`dexp(x, rate=1)`



$x \geq 0$

λ Abklingrate

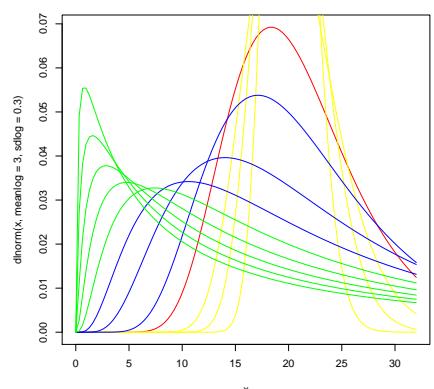
$$\mathcal{E}[\mathbb{X}] = \frac{1}{\lambda}$$

$$\text{Var}[\mathbb{X}] = \frac{1}{\lambda^2}$$

$$f_\lambda(x) = \lambda \cdot \exp \{-\lambda \cdot x\}$$

Lognormalverteilung

`dlnorm (x, meanlog=0, sdlog=1)`



$x \geq 0$

μ Mittel (Logskala)

σ^2 Varianz (Logskala)

$$\mathcal{E}[\mathbb{X}] = \exp(\mu + \sigma^2/2)$$

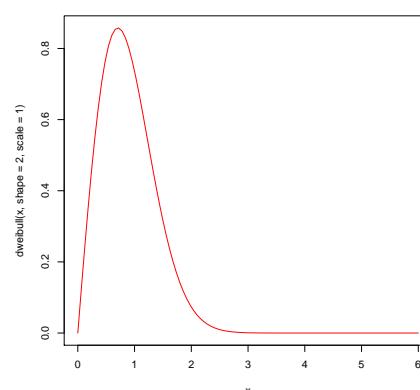
$$\text{Var}[\mathbb{X}] = \exp(2\mu + \sigma^2) \cdot (\exp(\sigma^2) - 1)$$

$$f_{\mu, \sigma}(x) = \frac{\mathcal{N}(\log x | \mu, \sigma)}{x} = \frac{1}{x \cdot \sigma \cdot \sqrt{2\pi}} \cdot \exp \left\{ -\frac{(\log x - \mu)^2}{2\sigma^2} \right\}$$

\mathbb{X} ist (μ, σ) -lognormalverteilt genau dann, wenn $\log(\mathbb{X}) \sim \mathcal{N}(\mu, \sigma)$

Weibullverteilung

`dweibull (x, shape, scale=1)`



$x \geq 0$

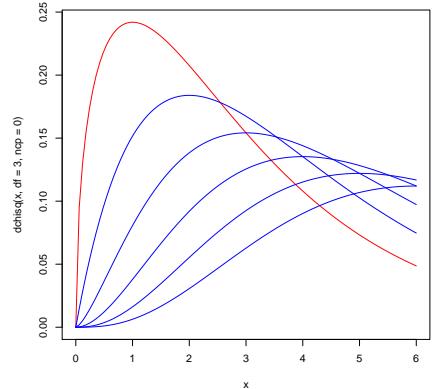
a Formparameter

s Skalenparameter

$$f_{a,s}(x) = \frac{a}{s} \cdot \left(\frac{x}{s}\right)^{a-1} \cdot \exp\left\{-\left(\frac{x}{s}\right)^a\right\}$$

Zentrale χ^2 -Verteilung

`dchisq (x, df)`



$x \geq 0$

n # Freiheitsgrade

$$\mathcal{E}[\mathbb{X}] = n$$

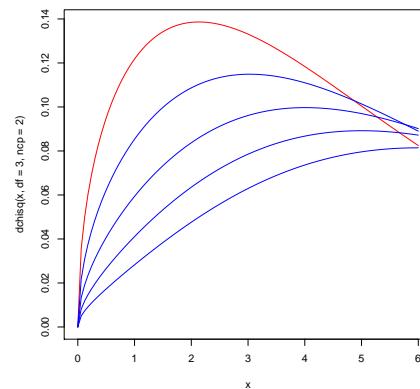
$$\text{Var}[\mathbb{X}] = 2n$$

$$f_n(x) = \chi_n^2(x) = \frac{1}{2^{n/2} \cdot \Gamma(d/2)} \cdot x^{n/2-1} \cdot e^{-x/2}$$

Summe der Quadrate von n vielen $\mathcal{N}(0, 1)$ -verteilten unabhängigen Zufallsvariablen

Nichtzentrale χ^2 -Verteilung

`dchisq (x, df, ncp=0)`



$x \geq 0$

n # Freiheitsgrade

λ Nicht-Zentralität

$$\mathcal{E}[\mathbb{X}] = n$$

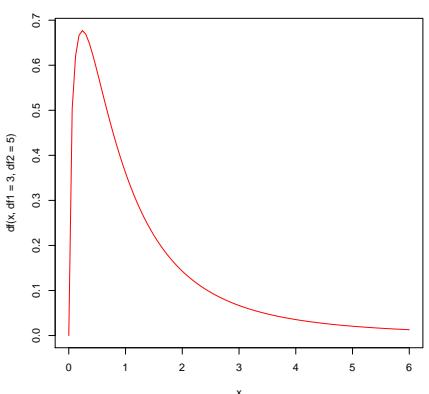
$$\text{Var}[\mathbb{X}] = 2n$$

$$f_{n,\lambda}(x) = \chi_{n,\lambda}^2(x) = e^{-\lambda/2} \cdot \sum_{r=0}^{\infty} \frac{(\lambda/2)^r}{r!} \cdot \chi_{n+2r}^2(x)$$

Summe der Quadrate von n vielen $\mathcal{N}(\lambda, 1)$ -verteilten unabhängigen Zufallsvariablen

F-Verteilung

`df (x, df1, df2)`



$$x \geq 0$$

n Freiheitsgrade Zähler

m Freiheitsgrade Nenner

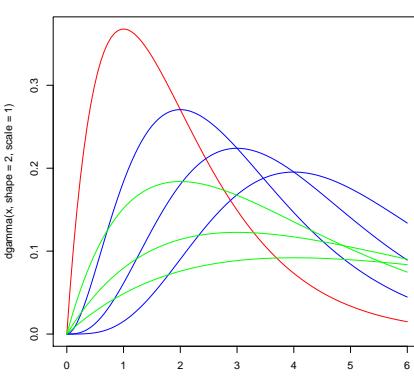
$$f_{n,m}(x) = \frac{\Gamma\left(\frac{n+m}{2}\right)}{\Gamma\left(\frac{n}{2}\right) \cdot \Gamma\left(\frac{m}{2}\right)} \cdot \left(\frac{n}{m}\right)^{n/2} \cdot x^{n/2-1} \cdot \left(1 + \frac{n}{m}x\right)^{-\frac{n+m}{2}}$$

Quotient der mittleren Quadratsummen von *n* bzw. *m* unabhängigen

$\mathcal{N}(0, 1)$ -verteilten Zufallsvariablen

Gammaverteilung

`dgamma (x, shape, rate=1, scale=1/rate)`



$$x \geq 0$$

a > 0 Formparameter

s > 0 Skalenparameter

r > 0 Rate $r = 1/s$

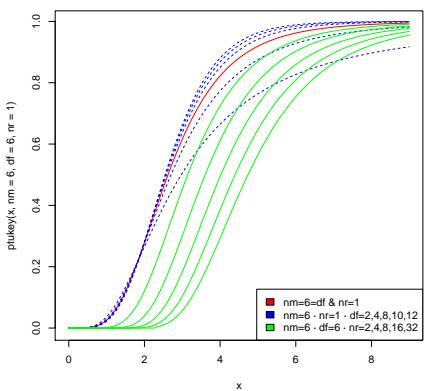
$$\mathbb{E}[X] = a \cdot s$$

$$\text{Var}[X] = a \cdot s^2$$

$$f_{a,s}(x) = \gamma_{a,s}(x) = \frac{1}{s^a \cdot \Gamma(a)} \cdot x^{a-1} \cdot e^{-x/s}$$

Tukeys studentisierte Spannenverteilung

`ptukey (q, nmeans, df, nranges=1, lower.tail=TRUE, log.p=FALSE)`



$$q \geq 0$$

nmeans Umfang *m* der $\mathcal{N}(0, 1)$ -Probe(n)

df Freiheitsgrade der Studentisierung

nranges Anzahl *n* gezogener Proben

$$F_{m,df,n}(x) \stackrel{\text{def}}{=} P(R/s \leq x)$$

R ist die maximale Spannweite unter *n* gezogenen $\mathcal{N}(0, 1)$ -Proben der Größe *m*. $df \cdot s^2$ ist (unabhängig davon) χ^2 -verteilt mit *df* Freiheitsgraden.

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf [0, 1]

Diskrete (univariate) Verteilungen

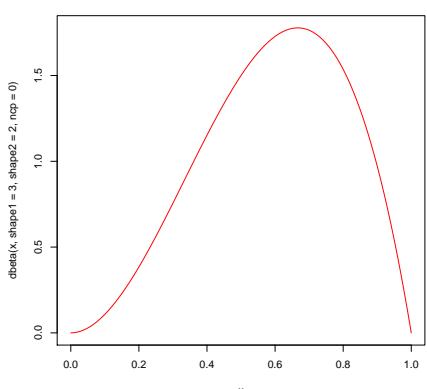
Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Betaverteilung

`dbeta (x, shape1, shape2, ncp=0)`



$$x \in [0, 1]$$

$a > 0$ Formparameter #1

$b > 0$ Formparameter #2

ncp Nicht-Zentralität

$$p_{a,b}(x) = \beta(x | a, b) = \frac{\Gamma(a+b)}{\Gamma(a) \cdot \Gamma(b)} \cdot x^{a-1} \cdot (1-x)^{b-1}$$

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

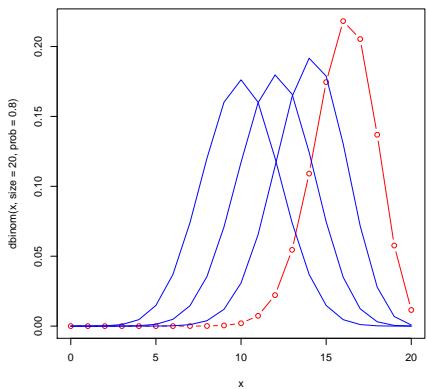
Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Binomialverteilung

`dbinom (x, size, prob)`



$$x \in \{0, 1, 2, \dots, n\}$$

n Bernoulli-Versuche

p Trefferwahrscheinlichkeit

$$\mathcal{E}[\mathbb{X}] = n \cdot p$$

$$\text{Var}[\mathbb{X}] = n \cdot p \cdot (1 - p)$$

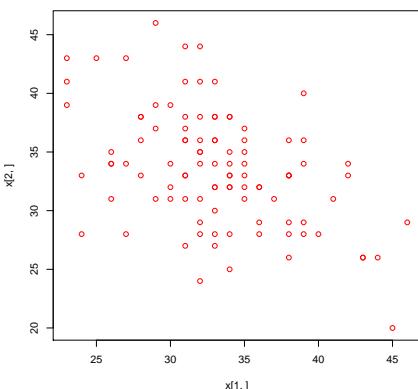
$$p_{n,p}(x) = \mathcal{B}(x | n, p) = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x}$$

Ziehen von n Kugeln mit Zurücklegen aus einer $(p, 1-p)$ -Urne

Multinomialverteilung

`dmultinom (x, size=sum(x), prob)`

$n=100$ Versuche aus $p=(1/3, 1/3, 1/3)$



$$x \in \mathbb{N}^K$$

$$n = \sum_k x_k \text{ Versuche}$$

$p \geq 0$ kanonische Parameter

$$\mathcal{E}[\mathbb{X}_k] = n \cdot p_k$$

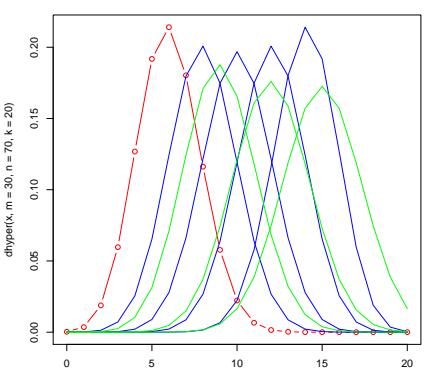
$$\text{Cov}[\mathbb{X}_i, \mathbb{X}_j] = -n \cdot p_i p_j$$

$$p_{n,p}(x) = \mathcal{B}(x | n, p) = \binom{n}{x} \cdot \prod_{k=1}^K p_k^{x_k} = \frac{(\sum_k x_k)!}{\prod_k x_k!} \cdot \prod_{k=1}^K p_k^{x_k}$$

Ziehen von n Kugeln mit Zurücklegen aus einer (p_1, \dots, p_K) -Urne

Hypergeometrische Verteilung

dhyper (x, m, n, k)



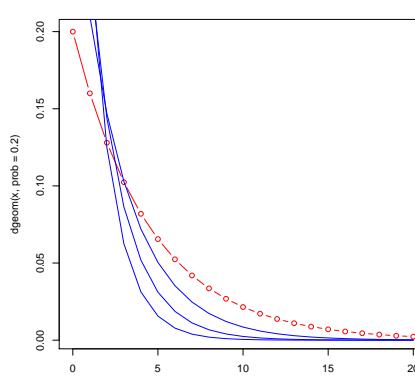
- $x \in \{0, 1, 2, \dots, k\}$
- m weiße Kugeln
- n schwarze Kugeln
- k gezogene Kugeln

$$p_{n,m,k}(x) = \frac{\binom{m}{x} \cdot \binom{n}{k-x}}{\binom{m+n}{k}}$$

Ziehen von k Kugeln **ohne** Zurücklegen aus einer (m, n) -Urne

Geometrische Verteilung

dgeom (x, prob)



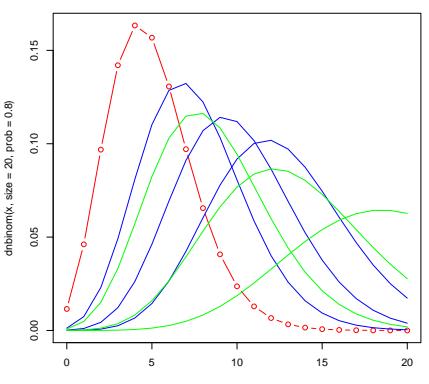
- $x \in \mathbb{N}$
- p Trefferwahrscheinlichkeit
- $E[\mathbb{X}] = \frac{(1-p)}{p}$
- $\text{Var}[\mathbb{X}] = \frac{(1-p)}{p^2}$

$$p_p(x) = p \cdot (1 - p)^x$$

Anzahl x der Fehlversuche bevor der erste Treffer gelandet wurde

Negative Binomialverteilung

dnbinom (x, size, prob, mu)



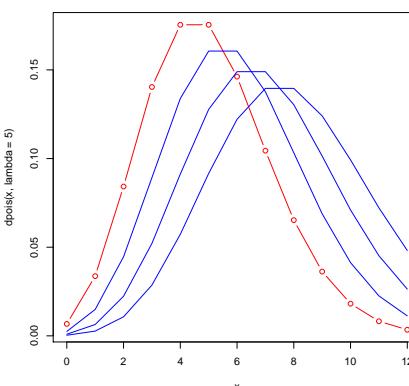
- $x \in \mathbb{N}$
- n Trefferzahl
- p Trefferwahrscheinlichkeit
- μ Erwartungswert (statt p)
- $E[\mathbb{X}] = \frac{n}{p} - n$
- $\text{Var}[\mathbb{X}] = \mu + \frac{\mu^2}{n}$

$$p_{n,p}(x) = \frac{\Gamma(x+n)}{\Gamma(n) \cdot x!} \cdot p^n \cdot (1-p)^x$$

Anzahl x der Fehlversuche bevor der n -te Treffer gelandet wurde

Poissonverteilung

dpois (x, lambda)



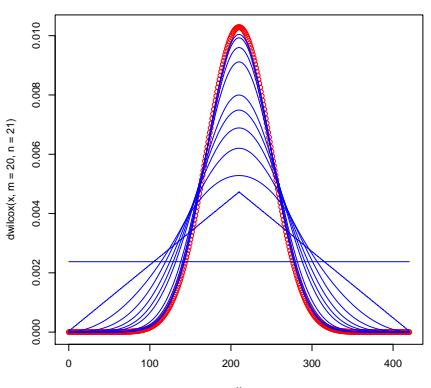
- $x \in \mathbb{N}$
- $\lambda > 0$ Formparameter
- $E[\mathbb{X}] = \lambda$
- $\text{Var}[\mathbb{X}] = \lambda$

$$p_\lambda(x) = \lambda^x \cdot \exp \left\{ -\frac{\lambda}{x!} \right\}$$

Anzahl x der Treffer eines seltenen ($p \rightarrow 0$) Ereignisses nach $n = \frac{1}{p}$ Versuchen

Wilcoxon Rangsummenverteilung

`dwilcox (x, m, n)`



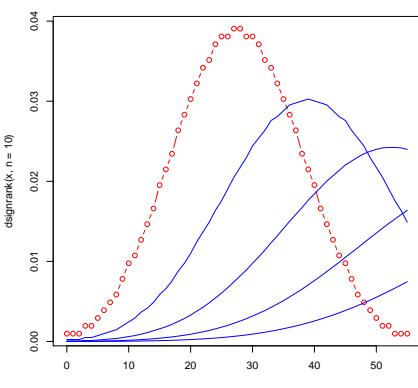
$$\begin{aligned} x &\in \{0, 1, 2, \dots, mn\} \\ m &\in \mathbb{N} \text{ Probenumfang} \\ n &\in \mathbb{N} \text{ Probenumfang} \\ \mathcal{E}[\mathbf{X}] &= \frac{mn}{2} \\ \text{Var}[\mathbf{X}] &= (m+n+1) \cdot \frac{mn}{12} \end{aligned}$$

$$p_{m,n}(x) = P\left(\sum_{i=1}^m \sum_{j=1}^n \mathbf{1}_{A_i \geq B_j} = x\right)$$

Für wieviele Paare (a_i, b_j) der i.i.d. Daten $\mathbf{a} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^n$ gilt $a_i \geq b_j$?

Wilcoxon Vorzeichen-Rangsummenverteilung

`dsignrank (x, n)`



$$\begin{aligned} x &\in \{0, 1, 2, \dots, (n+1) \cdot \frac{n}{2}\} \\ n &\in \mathbb{N} \text{ Probenumfang} \\ \mathcal{E}[\mathbf{X}] &= (n+1) \cdot \frac{n}{4} \\ \text{Var}[\mathbf{X}] &= (n+1)(2n+1) \cdot \frac{n^2}{24} \end{aligned}$$

$$p_n(x) = P\left(\sum_{i=1}^n \sum_{j=1}^n \mathbf{1}_{|A_i| \geq |B_j|} \cdot \mathbf{1}_{A_i > 0} = x\right)$$

Wie groß ist die Summe der **Absolutränge** aller positiven Zahlen a_i ?

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Daten, Wahrscheinlichkeiten und Vorhersage

Auswürfeln einer Stichprobe · Maximum-Likelihood-Schätzung

Wahrscheinlichkeitsmodell
Multivariate Verteilungsdichte

$$f : \begin{cases} \mathbb{R}^n & \rightarrow \mathbb{R}_0^+ \\ \mathbf{x} & \mapsto f(x_1, \dots, x_n) \end{cases}$$

Datensatz
Matrix (Objekte \times Merkmale)

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix}$$

Modell \Rightarrow Daten
Ziehen (ohne Zurücklegen)

$$f(\mathbf{X}) = \prod_{i=1}^m f(x_{i1}, \dots, x_{in})$$

i.i.d. $\hat{=} \text{independ. ident. distrib.}$

Daten \Rightarrow Modell
Maximale Datenerzeugungswahrscheinlichkeit

$$\hat{f}_{\text{ML}} = \underset{f \in \mathcal{M}}{\operatorname{argmax}} f(\mathbf{X})$$

\mathcal{M} Menge (parametrischer)
Verteilungskandidaten

Kettenregel und Vorhersagemodell

Kettenregel der Wahrscheinlichkeitstheorie

Produkt **bedingter** Wahrscheinlichkeiten (*a posteriori*)

$$f(x_1, \dots, x_n) = \prod_{j=1}^n f(x_j | x_1, \dots, x_{j-1})$$

multivariate Statistik \Rightarrow Prädiktion: $y \leftarrow x_1, \dots, x_n$

Regressionsanalyse

Wahrscheinlichster y -Wert

$$\hat{y}(x) = \mathcal{E}_{f(\mathbb{Y}|x)}[\mathbb{Y}]$$

bei gegebenen Quellvariablen

Bemerkung

Gleichwertig: **Regression** mit $(\mathbb{Y}, \mathbb{X}) \sim \mathcal{N}(\mu, \mathbf{S})$ und **OLS** mit $h(x, a) = a_0 + \sum_j a_j x_j$

Ausgleichsrechnung (OLS)

min. quadrat. Vorhersagefehler

$$\sum_{i=1}^m (h(x_i, a) - y_i)^2 \stackrel{!}{\rightarrow} \text{MIN}$$

Vorhersagemodelle

„Eierlegende Wollmilchsau“ für Mustererkennung, Data Mining & Künstliche Intelligenz

Interpolation von Funktionsverläufen

Stützstellenwerte $y_i = h(x_i)$ unbekannter funktionaler Abhängigkeiten $h(\cdot)$

Modellierung beobachteter Funktionswerte

Verrauschte Beispiele $y_i \approx h(x_i)$ unbekannter funktionaler Abhängigkeiten $h(\cdot)$

- + Glättung des Funktionsverlaufs
- + Korrelation und Abhängigkeiten höherer Ordnung
- + globales Datenmodell nach Kettenregel
- + Vorhersage zukünftiger Werte bei Zeitreihen (Extrapolation)

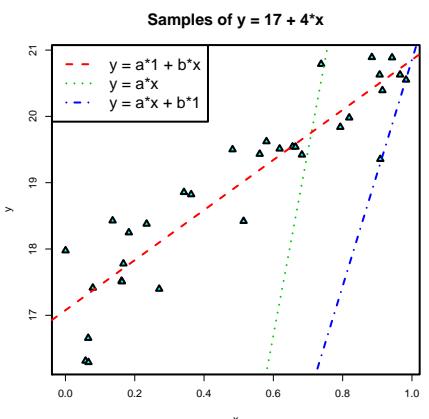
Klassifikation

$x_1, \dots, x_n \in \mathbb{R}^n$ Mustermerkmale, $y \in \{1 : K\}$ Klassenvariable

Lineares Vorhersagemodell

Linearkombination der Quellvariablen & Versatz („intercept“) in y -Richtung

$$\mathbb{Y}|x = h(x, a) + \mathbb{E} = a_0 + \sum_{j=1}^n a_j \cdot x_j + \mathbb{E} \quad \begin{cases} a_0 & h(\mathbf{0}, a) \\ a_j & x_j\text{-Steigung} \\ \mathbb{E} & \sim \mathcal{N}(0, \sigma^2) \end{cases}$$



Univariates OLS-Modell

Ausgleichsgerade $\hat{y} = a + b \cdot x$

```

y <- function (x, a=17, b=4) a+b*x
x <- matrix (runif (32), ncol=1)
y <- h(x) + rnorm (length(x), mean=0, sd=0.5)

plot (x, y, pch=24, bg="cyan",
      main="Samples of y = 17 + 4*x")

abline (reg=lm.fit (x, y), col=3, lty=3)
abline (reg=lm.fit (cbind(1,x), y), col=2, lty=2)
abline (reg=lm.fit (cbind(x,1), y), col=4, lty=4)

legend ("topleft", lty=2:4, col=2:4,
       cex=1.5, legend=leg.text)
  
```

Lineares OLS-Modell in R

`lm.fit (x, y, offset=NULL, method='qr', tol=1e-7, singular.ok=TRUE, ...)`

Beispielaufruf

`lm.fit (cbind (1, as.matrix (iris[-5])), iris$Species=='setosa')`
 ↵ Listenobjekt `lobj` mit Einträgen (unter anderem):

- `lobj$coefficients` Modellkoeffizienten a_j (je x-Spalte)
- `lobj$fitted.values` Vorhersagewerte $\hat{y}_i = \mathbf{a}^\top \mathbf{x}_i$ (je x-Zeile)
- `lobj$residuals` Soll-Ist-Differenzen $r_i = y_i - \hat{y}_i$ (je x-Zeile)

Problem

Umständliche Datenmatrixkonstruktion

Explizite (unökonomische) Expansion

Nichtnumerische Attributskalen

Wenig intuitive Aufrufsyntax

Spalten, 1=Versatz, Interaktionen?

Polynomterme!!

multiple Binärkodierung von Faktoren

Modellformeln

Kompakte Notation für Vorhersagemodelle

Syntax für Modellformeln

`<formula> ::= <response> ~ <explaining variables>`

- **Zielvariable (*response*)**

sind im Namensraum bekannte Datenvektoren

`x, 1:12, Sepal.Length, Species`

- **Menge erklärender Variablen (*explaining*)**

ist sprachlicher Ausdruck für eine Kombination von Termen

- **Vereinigungsoperator**

`Sepal.Length + Petal.Width + Species`

(die Konstante '1' ist immer implizit dabei)

- **Differenzoperator**

`Sepal.Length + Petal.Width - 1`

Lineare (Quadratmittel-)Modelle

`lm (formula, data, subset, weights, ...)`

- **Vorhersagemodell für verrauschte Daten**

`x <- 1:100`

`y <- 4*x + 17 + rnorm(length(x))`

Gesucht: die Koeffizienten des LSE-Modells

- **Berechnung eines affinen Modells**

`lm.aff <- lm (y ~ x)`

- **Berechnung eines linearen Modells**

`lm.lin <- lm (y ~ x-1)`

- **Berechnung mittels Datensatzvariablen**

`lm.iris <- lm (Petal.Width ~ Sepal.Width, data=iris)`

- **Alle außer Zielvariable**

`lm.iris <- lm (Petal.Width ~ ., data=iris[-5])`

Modellgetriebene Vorhersage

`predict.lm (object, newdata, ...)`

- **Vorhersage für die Lerndaten des Modells**

`guess <- predict (object=lm.aff)`
(polymorpher Aufruf mit `lm`-Objekt)

- **Vorhersage für frische Eingabedaten**

`guess <- predict (lm.aff, newdata=list(x=1:5))`
(Datensatz mit passenden Variablennamen!)

- **Vorhersage für Datensätze**

`guess <- predict (lm.iris, newdata=iris[1:50,-5])`

Interaktionsterme in Modellformeln

Mengenalgebra versus Variablenarithmetik

- **Elementare Termmengen**

Variablennamen, Intercept '1', Punktoperator '.'

- **Termmengenalgebra**

Plus '+', Minus '-', runde Klammern
 $(x+y+z)+(x+z)-z$
 x, y, z

- **Interaktionsterme**

alle paarweisen Produkte, keine Doubletten, keine Quadrate
 $(x+y+z):(x+z)$
 $x:z, y:x, y:z, 1$

- **Kumulative Interaktionsterme**

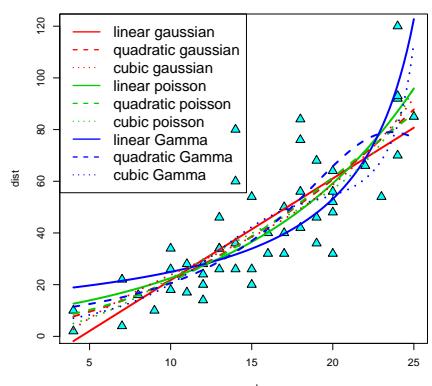
$(x+y+z)*(x+z)$
 $x, y, z, x:z, y:x, y:z, 1$

- **Potenzoperator**

$(x+y+z)^3$
 $1, x, y, z, x:y, x:z, y:z, x:y:z$

Beispiel — Bremsweg aus Geschwindigkeit schätzen

Datensatz 'cars' mit 50 Fällen (speed [mph], dist [ft]) aus den 1920ern



```
attach (cars)
plot (dist~speed, pch=24, bg="cyan", cex=1.4)

spd <- seq (min(speed), max(speed), len=100)
frms <- list (
  linear=dist~speed,
  quadratic=dist~speed + I(speed^2),
  cubic=dist~speed + I(speed^2) + I(speed^3)
)
fams <- list ("gaussian", "poisson", "Gamma")

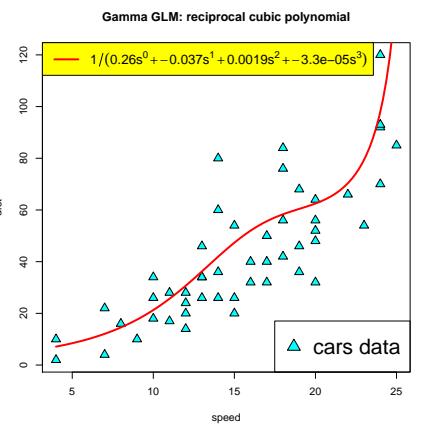
for (i in seq(along=frms))
  for (j in seq(along=fams)) {
    o <- glm (
      formula=frms[[i]],
      family=fams[[j]])
    guess <- predict (o,
      newdata=list (speed=spd),
      type="response")
    lines (spd, guess, lty=0+i, col=1+j, lwd=3)
  }
legend ("topleft",
  lty=rep (0+seq(along=frms), times=length(fams)),
  col=rep (1+seq(along=fams), each=length(frms)),
  legend=paste (
    rep (names(frms), times=length(fams)),
    rep (fams, each=length(frms))))
```

Vorhersagegüte

	gaussian	poisson	Gamma
linear	419.1	524.0	427.3
quadratic	418.7	515.6	418.6
cubic	419.8	506.1	412.4

Beispiel — das erfolgreichste (AIC) Bremsweg-Modell

Gamma-Verteilung · reziproke Linkfunktion · Polynom dritten Grades (in speed)



```
o <- glm (
  dist ~ speed+I(speed^2)+I(speed^3),
  family=Gamma, data=cars)
cf <- signif (coef(o), 2)
poly <- paste (cf, "*s^", 0:3, collapse="+")
body <- paste ("`1(`, poly, `")")
decl <- paste ("fun <- function(s)", body)
eval (parse (text=decl))
```

```
plot (dist ~ speed, data=cars,
  pch=24, bg="cyan", cex=1.4)
curve (fun, add=TRUE, lwd=3, col="red")
legend ("topleft",
  legend=body (fun),
  bg="yellow", lty=1, col="red")
```

coef (o)	(Intercept)	speed	I(speed^2)	I(speed^3)
	2.6037e-01	-3.6677e-02	1.8853e-03	-3.2829e-05
		2.5 %	97.5 %	
confint (o, level=0.95)	(Intercept)	speed	I(speed^2)	I(speed^3)
	1.544566e-01	-5.848412e-02	7.085221e-04	1.5161512e-03
		I(speed^2)	I(speed^3)	
		-5.691372e-05	-1.021716e-05	

Nichtlineare Modelle in R

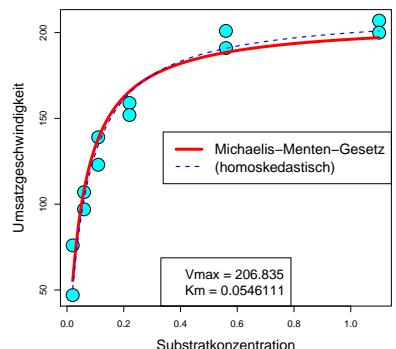
nls (formula, data, start, control, algorithm, subset, weights, ...)

Michaelis-Menten-Kinetik für Enzymreaktionen

Reaktionsgeschwindigkeit \downarrow Konzentration, Konstanten V_{\max}, K_m

$$y_{\text{rate}} = (V_{\max} \cdot x_{\text{con}}) / (K_m + x_{\text{con}}) \quad \text{und} \quad \sigma_y \propto \sqrt{y_{\text{rate}}}$$

Puromycin (Enzymreaktion)



```
Treated <- subset (Puromycin, state=="treated")
plot (rate ~ conc, data=Treated,
  pch=21, cex=3, bg="cyan")

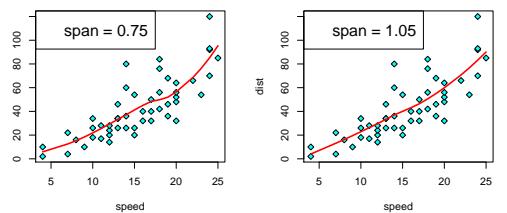
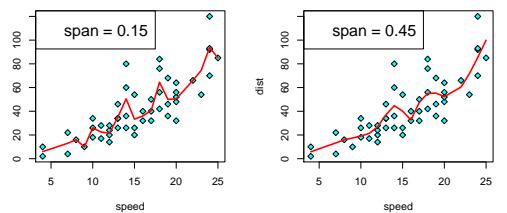
WMM <- function (rate, conc, Vmax, Km) {
  guess <- (Vmax * conc) / (Km + conc)
  (rate - guess) / sqrt (guess)
}

o <- nls (0 ~ WMM (rate, conc, Vmax, Km),
  data=Treated,
  start=list (Vmax=200, Km=0.1))

cf <- coef (o)
for (i in seq(along=cf))
  assign (names(cf)[i], cf[i])
curve ((Vmax*x)/(Km+x), add=TRUE, lwd=5, col="red")
```

Lokale polynomiale Regression

loess (formula, data, weights, subset, span=0.75, degree=2, ...)



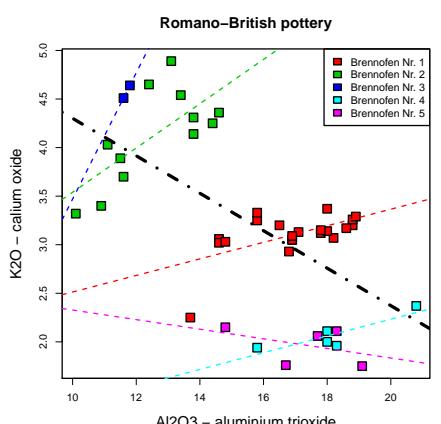
Locally WEighted Scatterplot Smoothing (Vanille-Variante):
lowess (x, y=NULL, f=2/3, iter=3, delta=0.01*diff(range(xy\$x[o])))

Lokale Regression

Polynomgrad 0,1,2 degree
Nachbarschaft span
Daten formula data
maximal 4 Quellvariable !

```
for (sp in seq (.15, 1.1, .3))
{
  plot (dist~speed,
    data=cars,
    pch=23, bg="cyan")
  lines (cars$speed,
    predict (loess (
      dist~speed,
      data=cars,
      span=sp)),
    lwd=2, col="red")
```

Faktoren als Quellattribute



```
library (HSAUR2); attach (pottery)
```

```
plot (K2O ~ Al2O3,
      bg=1+unclass (pottery$kiln))
abline (lm (K2O ~ Al2O3), lty=4, lwd=5)
for (i in seq(along=levels(kiln)))
  abline (lm (K2O ~ Al2O3,
             subset=unclass(kiln)==i), lty=2, col=1+i)
```

Pottery-Datensatz

45 Fundstücke

5 Brennöfen

9 Chemikalien:

kiln

A1203	aluminium trioxide
Fe2O3	iron trioxide
MgO	magnesium oxide
CaO	calcium oxide
Na2O	natrimum oxide
K2O	calium oxide
TiO2	titanium oxide
MnO	mangan oxide
BaO	barium oxide

Akaike-Information

Vorhersage von K_2O aus

Al_2O_3 allein	97.30
Al_2O_3 und kiln	16.11
kiln allein	29.11

Vorhersageformel für gemischte Attributskalen?

Faktoren und Vorhersageformeln

Faktorattribut $k \in \{1, 2, \dots, \ell\}$ \Rightarrow numerische Hilfsattribute $k_1, \dots, k_\ell \in \mathbb{R}$

Beispiel Al_2O_3 -Vorhersage (pottery-Daten)

$y =$	K20	kiln	K20+kiln	K20:kiln	K20*kiln
1· Intercept	21.9	16.9	8.68	6.04	4.44
+ x· K20	-1.94		2.66		4.02
+ $k_2 \cdot$ kiln2		-4.36	-7.07		-0.93
+ $k_3 \cdot$ kiln3			-5.22	-9.13	0.23
+ $k_4 \cdot$ kiln4				1.26	3.99
+ $k_5 \cdot$ kiln5				0.40	3.42
+ $x \cdot k_1 \cdot$ K20:kiln1					3.50
+ $x \cdot k_2 \cdot$ K20:kiln2					1.59
+ $x \cdot k_3 \cdot$ K20:kiln3					1.24
+ $x \cdot k_4 \cdot$ K20:kiln4					5.86
+ $x \cdot k_5 \cdot$ K20:kiln5					5.66
AIC	201.1	173.8	160.8	162.3	159.4

Bemerkung

Überflüssige Hilfsvariable (lineare Abhängigkeit!) werden automatisch getilgt.

Welche Werte bekommen die Hilfsvariablen?

```
model.matrix (object, data=environment(object), contrasts.arg=NULL, ...)
```

Ohne Interaktionen zwischen kiln und K20

```
model.matrix (Al2O3~K20+kiln, pottery)
```

	(Intercept)	K20	kiln2	kiln3	kiln4	kiln5
1	1	3.20	0	0	0	0
2	1	3.05	0	0	0	0
3	1	3.07	0	0	0	0
22	1	4.25	1	0	0	0
23	1	4.14	1	0	0	0
34	1	4.51	0	1	0	0
36	1	1.96	0	0	1	0
41	1	2.06	0	0	0	1
45	1	1.75	0	0	0	1

Mit Interaktionen zwischen kiln und K20

```
model.matrix (Al2O3~K20:kiln, pottery)
```

	(Intercept)	K20:kiln1	K20:kiln2	K20:kiln3	K20:kiln4	K20:kiln5
1	1	3.20	0.00	0.00	0.00	0.00
2	1	3.05	0.00	0.00	0.00	0.00
3	1	3.07	0.00	0.00	0.00	0.00
22	1	0.00	4.25	0.00	0.00	0.00
23	1	0.00	4.14	0.00	0.00	0.00
34	1	0.00	0.00	4.51	0.00	0.00
36	1	0.00	0.00	0.00	1.96	0.00
41	1	0.00	0.00	0.00	0.00	2.06
45	1	0.00	0.00	0.00	0.00	1.75

Faktoren und Kontrastmatrizen

contrasts (x, contrasts=TRUE, sparse=FALSE) — zum Abfragen und Setzen

contrasts (pottery\$kiln)

	2	3	4	5
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1

contrasts (factor (LETTERS[1:6]))

	B	C	D	E	F
A	0	0	0	0	0
B	1	0	0	0	0
C	0	1	0	0	0
D	0	0	1	0	0
E	0	0	0	1	0
F	0	0	0	0	1

contrasts (Windrose) <- "contr.treatment"

	west	sued	ost
nord	0	0	0
west	1	0	0
sued	0	1	0
ost	0	0	1

contrasts (Windrose) <- "contr.helmert"

	[,1]	[,2]	[,3]
nord	-1	-1	-1
west	+1	-1	-1
sued	0	+2	-1
ost	0	0	+3

contrasts (Windrose) <- "contr.sum"

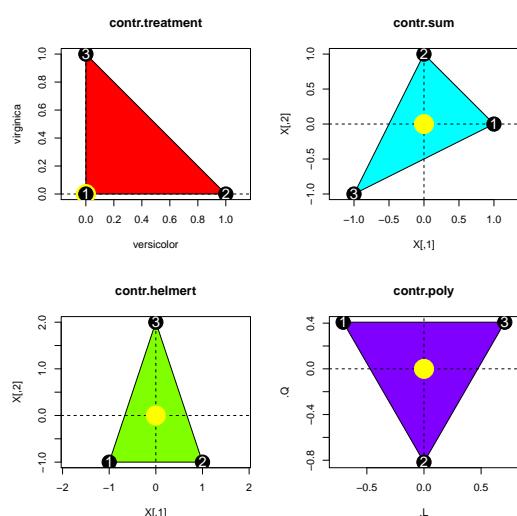
	[,1]	[,2]	[,3]
nord	1	0	0
west	0	1	0
sued	0	0	1
ost	-1	-1	-1

contrasts (Windrose) <- "contr.poly"

	.L	.Q	.C
nord	-0.6708204	0.5	-0.2236068
west	-0.2236068	-0.5	0.6708204
sued	0.2236068	-0.5	-0.6708204
ost	0.6708204	0.5	0.2236068

Geometrie von ℓ Punkten des $\mathbb{R}^{\ell-1}$

Äquidistante Codekonfiguration durch orthogonale Polynome



```
contr <- paste ("contr", c(
  "treatment",
  "helmert",
  "sum",
  "poly"
), sep=".")
attach (iris)
layout (matrix (1:4, 2, 2))
for (i in seq(along=contr)) {
  contrasts (Species) <- contr[i]
  X <- contrasts (Species)
  plot (X, asp=1, main=contr[i])
  polygon (X, col=rainbow(4)[i])
  abline (h=0, v=0, lty=2)
  points (0, 0,
    cex=4, col="yellow", pch=19)
  points (X, pch=19, cex=3)
  text (X, labels=1:3,
    col="white", cex=1.4)
}
```

Faktoren als Zielattribute

Verwendung von Vorhersagemodellen zu Klassifikationszwecken

- Faktoren können nicht Zielvariable sein!**

```
lm (Species ~ ., data=iris) Fehler in storage.mode(y) <- "double" ...
```

- Kontrastmatrix basteln**

```
Y <- diag (length (levels (Species))) Einheitsvektoren ( $\mathbb{R}^3$ )
Y <- contr.helmert (levels (Species)) Helmertkontraste ( $\mathbb{R}^2$ )
```

- Zielmatrix erzeugen**

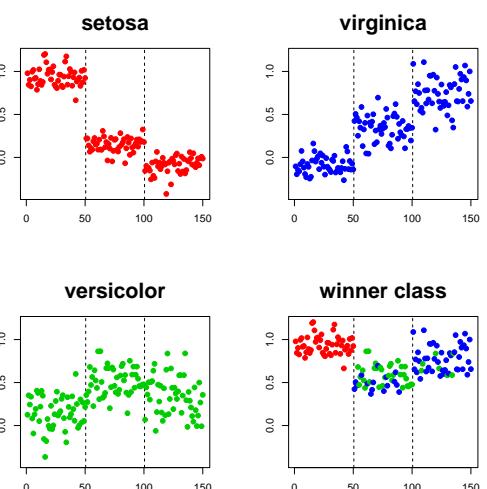
```
y <- Y[Species,] je Faktoreintrag passende Kontrastzeile ( $\mathbb{R}^{150 \times 3}$ )
```

- Multiples Vorhersagemodell**

	[,1]	[,2]	[,3]
(Intercept)	0.118223	1.577059	-0.695282
Sepal.Length	0.066030	-0.020154	-0.045876
Sepal.Width	0.242848	-0.445616	0.202768
Petal.Length	-0.224657	0.220669	0.003988
Petal.Width	-0.057473	-0.494307	0.551779

Beispiel — lineare OLS-Klassifikation

$$150 \text{ Irisblüten} = 50 \times \text{setosa} + 50 \times \text{versicolor} + 50 \times \text{virginica}$$



```
attach (iris)
y <- diag(3)[Species,]
o <- lm (y~.-Species, data=iris)
u <- predict (o, newdata=iris)

par (pch=19, cex.main=2)
layout (matrix (1:4, 2, 2))

newplot <- function (S, brx=NULL, ...) {
  plot (c(1,nrow(S)), range(S),
    xlab="", ylab="", type="n", ...)
  abline (v=1/2+brx, lty=2)
}

clab <- levels (Species)
for (k in seq(along=clab)) {
  newplot (u, c(50,100), main=clab[k])
  points (u[,k], col=1+k)
}

u.max <- apply (u, MARGIN=1, max)
k.max <- apply (u, MARGIN=1, which.max)
newplot (u, c(50,100), main="winner class")
points (u.max, col=1+k.max)
```

Weitere Vorhersagemodelle

- Korrelierte Vorhersagefehler**

Generalized least squares

```
nlme::gls()
```

- Effekte von Quellvariablen auf Parameter**

Linear mixed-effects model

```
nlme::lme()
```

Non-linear mixed-effects model

```
nlme::nlme()
```

- Verallgemeinerte lineare Modelle**

Splineregression, automatische Glättung

```
mgcv::gam()
```

- Vorhersage mit neuronalen Netzen**

Single hidden layer perceptron

```
nnet::nnet()
```

- Vorhersage geordneter Faktoren**

Proportional-odds logistic regression

```
MASS::polr()
```

Bemerkung

Mehr Modelle zur Vorhersage von Faktoren im Kapitel zur „Klassifikation“

Funktionen für Wahrscheinlichkeitsverteilungen

Generelle Statistikfunktionen

Stetige univariate Verteilungen

Nichtnegative univariate Verteilungen

Univariate Verteilungen auf $[0, 1]$

Diskrete (univariate) Verteilungen

Formelschnittstelle für Vorhersagemodelle

Lineare und nichtlineare Modelle

Numerische Optimierung

Was ist Optimierung?

Spezielle Form der inversen Aufgabenstellung zur Funktionsauswertung

$$f : \begin{cases} \mathcal{M} & \rightarrow \mathbb{R} \\ x & \mapsto f(x) \end{cases} \quad \begin{array}{l} f(x_{\min}^*) \leq f(x) \quad (\forall x \in \mathcal{M}) \\ f(x_{\max}^*) \geq f(x) \quad (\forall x \in \mathcal{M}) \end{array}$$

Suchraum $\hat{=}$ Definitionsbereich

Diskrete/kombinatorische Optimierung

Skalare (univariate) Optimierung

Mehrdimensionale (multivariate) Optimierung

$|\mathcal{M}| < \infty$

$\mathcal{M} = \mathbb{R}$

$\mathcal{M} = \mathbb{R}^n$

Suchstrategie

Traversieren des Raums

Geordnete Suche (Kand.-Liste)

Evolutionäre Suche

globales vs. lokales Optimum

Suchinformation

Funktionsauswertung $f(x)$

Gradientenauf/abstieg $f'(x)$

Schrittweitenbestimmung $f''(x)$

deterministisch vs. stochastisch

Skalare Optimierung (goldener Schnitt)

`optimize (f=, interval=, ..., lower, upper, maximum=FALSE)`

- **Minimum einer Parabel**

```
optimize (f=function(r) (r-3)^2, interval=c(-5,+5)) {min=3  
obj=0}
```

- **Maximum der Sinuswelle**

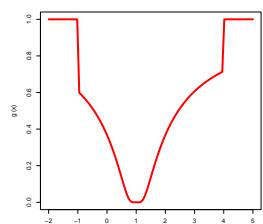
```
optimize (f=sin, interval=c(0,5), maximum=TRUE) {min=1.5708  
obj=1.0000}
```

- **Besetzen überschießender Funktionsargumente**

```
optimize (f=get('*'), interval=c(-5,+5), e2=2) {min=-4.999  
obj=-9.999}  
(Abholen des Funktionsobjekts; Namen der Argumente)
```

- **Wir legen den goldenen Schnitt herein!**

```
g <- function(x) ifelse (  
  x>-1, ifelse (  
    x<4, exp (-1/abs(x-1)), 1), 1)  
  
optimize(g, c(-7,20)) {min=0.99  
obj=0.000}  
optimize(g, c(-4,20)) {min=19.99  
obj=1.000}
```



Beispiel — Maximum-Likelihood-Schätzung

Optimale Parameter μ, σ einer normalverteilten Datenprobe

- **Erzeuge univariate Datenprobe**

```
x <- rnorm (100, mean=13, sd=3)
```

- **Definiere Likelihoodfunktion**

```
like <- function (mu=0, sd=1, data=x)  
  sum (dnorm (data, mu, sd, log=TRUE))
```

- **Maximiere hinsichtlich μ**

```
optimize (like, c(0,100), maximum=TRUE)  
maximum=12.9195 objective=-699.9
```

- **Maximiere hinsichtlich σ**

```
optimize (function(s) -like (sd=s), c(0,100))  
minimum=13.38 objective=401.3
```

- **Verwende geschätzten μ -Wert**

```
optimize (function(s) -like (mu=12.9, sd=s), c(0,100))  
minimum=3.48 objective=266.8
```

Univariate Nullstellensuche

```
uniroot (f, interval, ..., lower, upper, f.lower, f.upper, maxiter=1000)
```

- **Intervallangabe erforderlich**

```
uniroot (f=cos, interval=c(-1,+2))
root=1.57 f.root=1.2e-07 iter=5
```

- **Intervallgrenzen nur mit Vorzeichenwechsel!**

```
uniroot (f=cos, lower=-1, upper=+1)
f() values at end points not of opposite sign
```

- **Nur eine Nullstelle wird gesucht**

```
uniroot (function(x) (x-2)*(x-4), c(0,3))
root=2 f.root=-1.57e-05 iter=8
```

- **Überschließende Funktionsargumente**

```
uniroot (function(x,z) x*x-z, c(1,2), z=2)
root=1.41 f.root=-6.86e-07 iter=5
```

- **Argumentnamen aus Fehlermeldung**

```
uniroot (f='-', c(-1,+5), a=3)
root=3 f.root=0 iter=1
```

Multivariate Optimierung — Newton-Verfahren

```
nlm (f, p, ..., hessian=FALSE, gradtol=1e-6, iterlim=100)
```

- **Erzeuge univariate Datenprobe**

```
x <- rnorm (100, mean=13, sd=3)
```

- **Definiere negative Likelihoodfunktion**

```
neglike <- function (para, data=x)
-sum (dnorm (data, para[1], para[2], log=TRUE))
```

- **Minimiere simultan hinsichtlich para = (μ, σ)**

```
nlm (f=neglike, p=c(10,2)
minimum=250.5011 estimate=12.999695/2.962615 code=1 iterations=28
```

Newton-Abstieg benötigt Gradient & Hessematrix

implizit: $\nabla_p f$ und $H_p f$ werden von `nlm` numerisch angenähert.

explizit: Berechnungsfunktionen werden als Attribute `gradient` und `hessian` von `f` übergeben.

(Komplexe) Polynomwurzeln (Jenkins & Traub 1972)

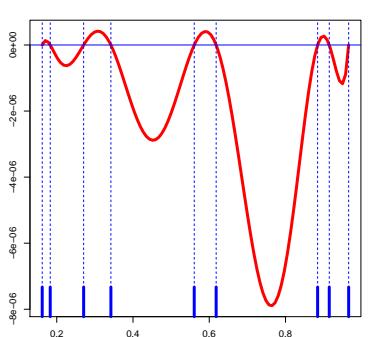
`polyroot (z)` mit $p(x) = z_1 + z_2 \cdot x + z_3 \cdot x^2 + \dots + z_n \cdot x^{n-1}$

- **Quadratisches Polynom**

```
polyroot (c(1,-3,1))
polyroot (c(1,0,1))
```

- **Vielfache Nullstellen**

```
choose(8, 0:8)
polyroot (choose(8, 0:8))
```



```
a <- runif (9)
poly <- function (x)
  apply (outer(x,a,"-"), 1, prod)
plot (poly, min(a), max(a),
  col="red", lwd=5)
abline (v=a, lty=2, col="blue")
abline (h=0, lty=1, col="blue")
rug (
  Re (polyroot (polyCF (a))),
  col="blue", lwd=5,
  ticksize=0.1)
```

Multivariate Minimierung mit Nebenbedingungen

```
nlminb (start, objective, gradient, hessian, ..., lower=-Inf, upper=Inf)
```

- **Maximum-Likelihood (wie nlm)**

```
nlminb (start=c(0,1), objective=neglike)
par=12.9997/2.9626 obj=250.501 iter=16 eval/fun=20 eval/grad=46
```

- **Maximale Spannweite gesucht**

```
foo <- function (z) -abs (diff (range (z)))
```

- **Pathologisch ohne Schranken:**

```
nlm (f=foo, p=1:5)      -26221.22 2.00 3.00 4.00 +26227.22
nlminb (start=1:5, objective=foo)
-5.1111e+11 2.0000e+00 3.0000e+00 4.0000e+00 5.1111e+11
```

- **Wohldefiniert mit Schranken:**

```
nlminb (start=1:5, obj=foo, lower=-883, upper=+4711)
-883 2 3 4 4711
```

- **Fokussierung bei relativen Minima**

```
foo <- function(z) sin(z[1])+cos(z[2])      f(z) = sin(z1) + cos(z2)
nlminb (c(35,35), foo, lo=33, up=37)
par=36.128/34.557 obj=-2
```

Multivariater Allzweckminimierer

```
optim (par, fn, gr=NULL, ..., method, lower, upper, control=list())
```

- Rosenbrocks Bananenfunktion

```
frb <- function(x)
  100 * (x[2]-x[1]^2)^2 + (1-x[1])^2
```

- ... und ihre ersten Ableitungen

```
grb <- function(x) c(
  -400 * x[1] * (x[2]-x[1]^2) - 2 * (1-x[1]),
  200 * (x[2]-x[1]^2))
```

- Kein Erfolg mit Nelder-Mead

```
optim (c(-1.2,1), frb)
  par=0.9998044/0.9996084 value=3.827383e-08
```

- BFGS mit Gradienten

```
optim (c(-1.2,1), frb, grb, method='BFGS')
  par=1/1 value=9.594956e-18
```

Methoden

Nelder-Mead

nur f-Werte

BFGS

Broyden,
Fletcher,
Goldfarb &
Shanno; f und f'

CG

Konjugierte
Gradienten
(HDim)

SANN

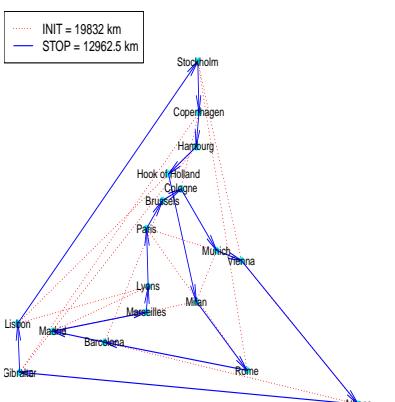
Simulated
Annealing

BFGS

BFGS +
Schranken

Travelling Salesperson Problem (TSP)

Näherungslösung durch „Simulated Annealing“



Bemerkung

Die SANN-Methode von optim erwartet als Argument gr keine Funktion für den Gradienten, sondern für den Folgekandidaten.

```
distance <- function (s, P=xy)
  sum (as.matrix (dist (P))[embed(s,2)])

newseq <- function (s, P=xy) {
  idx <- 3:nrow(P) - 1
  changepoints <- sample (idx, size=2)
  s[changepoints] <- rev (s[changepoints])
  s}

idx <- 1:nrow(xy)
s <- c (idx, 1)
tsp <- xy[s,]
plot (xy, asp=1,
  axes=FALSE, pch=19, col="cyan")
arrows (tsp[idx,1], tsp[idx,2],
  tsp[idx+1,1], tsp[idx+1,2],
  lty=3, length=0, angle=10, col="red")

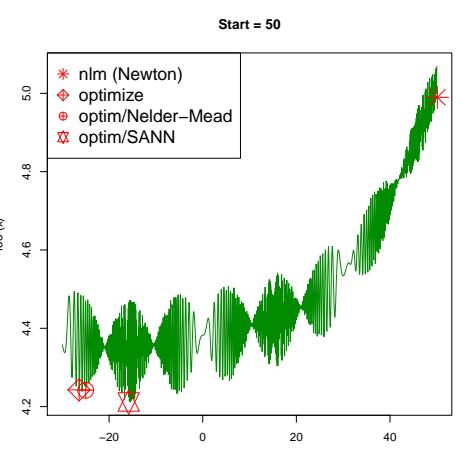
set.seed (883)
o <- optim (s, distance, newseq, method="SANN")
tsp <- xy[o$par,]
arrows (tsp[idx,1], tsp[idx,2],
  tsp[idx+1,1], tsp[idx+1,2],
  lty=1, length=0.2, angle=10, col="blue")

text (xy, rownames (xy), cex=0.8)
```

Skalare Optimierung — viele lokale Minima

Simulated Annealing mit 10000 Auswertungen und handverlesener Abkühlpolitik

$$f(x) = \log \left\{ 10 \cdot \sin(0.3x) \cdot \sin(1.3x^2) + \frac{x^4}{10^5} + \frac{x}{5} + 80 \right\}$$



```
foo <- function (x) log (
  10*sin(0.3*x)*sin(1.3*x^2) +
  0.00001*x^4 + 0.2*x + 80
)
plot (foo, -30, +50, n=1000,
  col="green4", main="Start = 50")

o <- nlm (f=foo, p=50)
points (o$estimate, o$minimum,
  col="red", pch=8, cex=3)
o <- optimize (f=foo, interval=c(-50,+50))
points (o$minimum, o$objective,
  col="red", pch=9, cex=3)
o <- optim (par=50, fn=foo, method="Nelder-Mead")
points (o$par, o$value,
  col="red", pch=10, cex=3)
o <- optim (par=50, fn=foo, method="SANN",
  control=list(parscale=20))
points (o$par, o$value,
  col="red", pch=11, cex=3)
```

Zusammenfassung (4)

- Mit **mean**, **var** etc. berechnen wir (getrimmte) **Mittel** und **Streuungen** bzw. **Kovarianzen** von Datenproben.
- Rangordnungen** werden mit **sort**, **rank**, **order** analysiert, **Quantile** mit **quantile** und **cut**.
- Verteilungen von Datenproben werden mit **qqplot** verglichen und mit **qnorm** auf **Normalität** getestet.
- Für alle gängigen Wahrscheinlichkeitsmodelle bietet 'R' Methoden [**d,p,q,r**]xxx für den **Dichte**verlauf, für die **kumulative Verteilung**, für die **Quantilberechnung** und für ein zufallsgesteuertes **Auswürfeln**.
- Modellformeln** der Art $V_{\text{resp}} \sim V_{\text{expl}}$ bieten eine kompakte mengentheoretische Notation (+, -, :, *, ^) für die **Interaktionsterme** statistischer **Vorhersagemodelle**.
- Die Modelle liefern **Schätzwerte** und **Konfidenzintervalle** ihrer Parameter, Qualitätsaussagen wie **AIC**, **ANOVA** und **Residuen** und sie unterstützen die **induktive Prädiktion**.
- Die Modelle sind **linear** (**lm**), **gekoppelt linear** (**glm**), **nichtlinear** (**nls**) oder **nichtparametrisch** (**loess**).
- Diskrete** Quell- oder Zielvariable („*Faktoren*“) werden mittels **Kontrastmatrizen** konvertiert.
- Optimierungsaufgaben** werden mit **optimize** (univariat), **uniroot** und **polyroot** (Nullstellen), **nlm[inb]** (Newton multivariat) oder **optim** (zerklüftet, kombinatorisch) gelöst.

WERKZEUGE DER MUSTERERKENNTUNG UND DES MASCHINELLEN LERNENS

Vorlesung im Sommersemester 2018

Prof. E.G. Schukat-Talamazzini

Stand: 19. Februar 2018

Teil V

Einige Highlights aus der reichhaltigen
Werkzeugkiste der Sprache 

Signalverarbeitung Transformation Regression Klassifikation Gruppierung

Implementationsschema für (etliche) -Modelle

Klasse: `mod` · Konstruktor: `mod()` · Methoden: `predict()`, `coef()`, ...

Standardschnittstelle

```
mod [.fit] (x, y, ...)
x   Quellvariablen: matrix/data.frame
y   Zielvariablen: matrix/vector
grouping   statt 'y': factor
```

Formelschnittstelle

```
mod (formula, data, ...)
formula   Modell tgt ~ src
data      Datensatz: data.frame
... weitere Aufrufargumente:
weights   Fehlergewicht: numeric
subset    Datenauswahl: logical
na.action Defekte (NA): function
```

Vorhersagemethode

```
predict (x, newdata, ...)
x, object gefittetes Modell: mod
newdata   Datensatz: data.frame
type     {
  n, q-Matrix: 'response'
  n, k-Matrix: 'probs'
  Faktor: 'class'
  n, p*-Matrix: 'terms'
  Vektor: 'link'}
```

Diverse Modellkomponenten

```
coef(x)   gelernte Modellparameter
fitted(x) Vorhersage für Lerndaten
residuals(x) Abweichung dabei
AIC(x)    Akaiques Modellgütewert
summary(x) Modellüberblick
... einige weitere generische Methoden:
print(x), plot(x), str(x), ...
```

Signalverarbeitung Transformation Regression Klassifikation Gruppierung

Digitale Signalverarbeitung

Merkmaltransformation und Dimensionsreduktion

Numerische und ordinale Vorhersage

Klassifikation und überwachtes Lernen

Clusteranalyse und unüberwachtes Lernen

Digitale Signalverarbeitung

Theorie: $\tilde{f} \hat{=} [\dots, f_{n-1}, f_n, f_{n+1}, \dots] \in \mathbb{R}^{\mathbb{Z}}$ · Praxis: $f \hat{=} (f_1, \dots, f_N)^T \in \mathbb{R}^N$

Repräsentation (Zeit)

Vektor **numeric**

Zeitreihe(n) **ts**, **mts**

LSI-Systeme $\tilde{f} \mapsto \tilde{h}$

Faltung $\tilde{h} = \tilde{f} * \tilde{g}$

mit Impulsantwort $\tilde{g} \in \mathbb{R}^{\mathbb{Z}}$

ARMA-Systeme (p, q)

$$h_n = \underbrace{\sum_{j=0}^q b_j \cdot f_{n-j}}_{\text{moving average}} - \underbrace{\sum_{j=1}^p a_j \cdot h_{n-j}}_{\text{autoregressive}}$$

Repräsentation (Frequenz)

Vektor **complex**

Diskrete Fouriertransform.

$$F_\nu = \frac{1}{N} \cdot \sum_{n=0}^{N-1} f_n \cdot e^{-2\pi \frac{\nu n}{N}}$$

für $\nu = 0, 1, 2, \dots, N-1$ bzw. $N/2$

Faltungssätze (LSI, ARMA, ACF)

$$H_\nu = F_\nu \cdot G_\nu$$

$$H_\nu = F_\nu \cdot B_\nu / A_\nu$$

$$R_\nu^f = |F_\nu|^2 \quad \text{wegen } r^f = f * f$$

'R'-Klassen für Zeitreihen: ts & mts

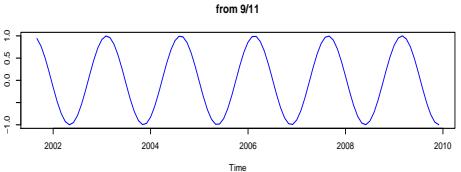
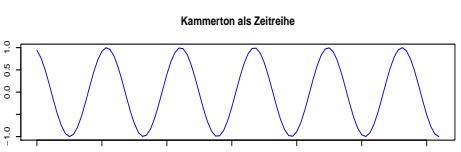
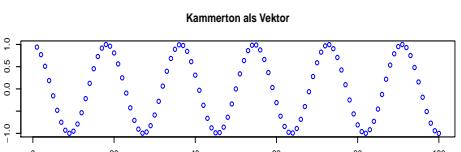
`stats:::ts (data=NA, start=1, end=, frequency=1, deltat=1, class=, names=)`

Funktionsargumente

data	Vektor/Matrix
start	Index(-paar)
end	Index(-paar)
frequency	#Observ./Einheit
deltat	(reziprok dazu)
names	multiple Zeitreihen

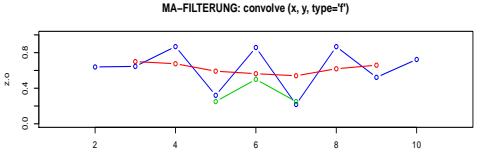
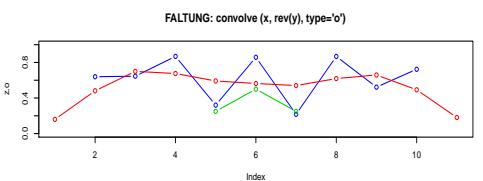
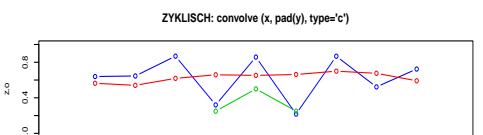
R-Programmcode

```
layout (1:3)
sp <- 1:100/8000
x <- cos (2*pi*sp*440)
s <- ts (x, start=0, freq=8000)
plot (x, col="blue", main=
      "Kammerton als Vektor")
plot (s, col="blue", main=
      "Kammerton als Zeitreihe")
plot (ts (
      x, freq=12, start=c(2001,9),
      col="blue", main="from 9/11")
```



Faltungsoperation mit Hilfe der FFT

`stats:::convolve (x, y, conj=TRUE, type=c('circular', 'open', 'filter'))`



Funktionsargumente

x	Vektor Nr.1
y	Vektor Nr.2
conj	komplex konjugiert?
type	Periode/Nullen
return	Vektor $x * y$

Filtern versus Falten

Implement. Faltung mit `rev(y)`:

$$z_i \stackrel{\text{def}}{=} \sum_{j \in \mathbb{Z}} x_j \cdot y_{j-i+|y|}$$

'c' periodisch fortsetzen

$|x|$ Koeffizienten, $|x| = |y|$

'o' mit 0 auffüllen: Faltmodus

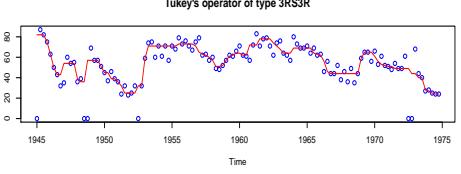
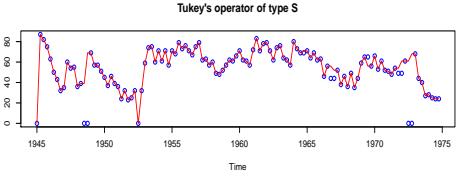
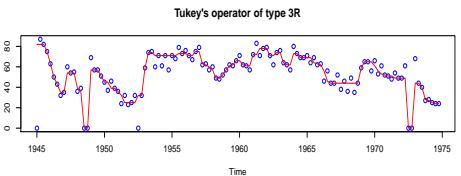
$|x| + |y| - 1$ Koeffizienten

'f' mit 0 auffüllen: Filtermodus

$|x| - |y| + 1$ Koeffizienten

Tukeys nichtlineare Glättungsoperatoren

`stats:::smooth (x, kind = c('3RS3R', '3RSS', '3RSR', '3R', '3', 'S'), twiceit=FALSE, endrule=c('Tukey', 'copy'), do.ends=FALSE)`



Funktionsargumente

x	Vektor oder Zeitreihe
kind	Operatortypcode
twiceit	doppelt gemoppt
endrule	Verhalten am Rand
return	Vektor oder Zeitreihe

Typcode-Konventionen

3: Median der Ordnung 3

S: Spalte alle 2/3-Läufe

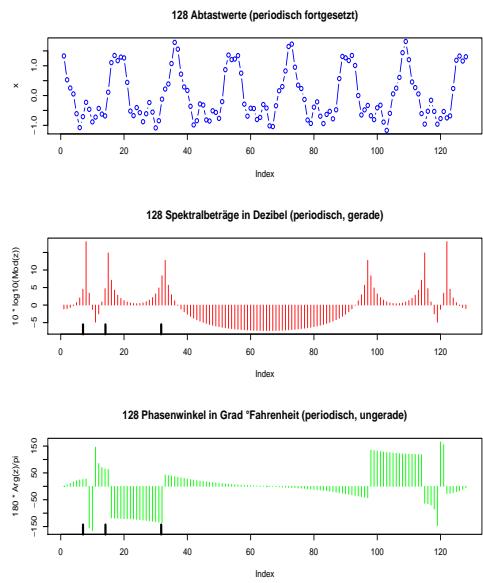
R: Wiederhole bis Konvergenz

R-Programmcode

```
x <- presidents
x[is.na(x)] <- 0
for (w in c("3R", "S", "3RS3R")) {
  plot (x, col="blue", type="p")
  lines (smooth (x, w), col="red")
}
```

(Schnelle) Fouriertransformation DFT : $\mathbb{C}^N \rightarrow \mathbb{C}^N$

stats:::mvfft (z, inverse=FALSE)



Funktionsargumente

z Vektor/Array über \mathbb{R}/\mathbb{C}
inverse rechne $N \cdot \text{DFT}^{-1}\{z\}$
return Vektor/Array über \mathbb{R}/\mathbb{C}

Schnelle diskrete FT

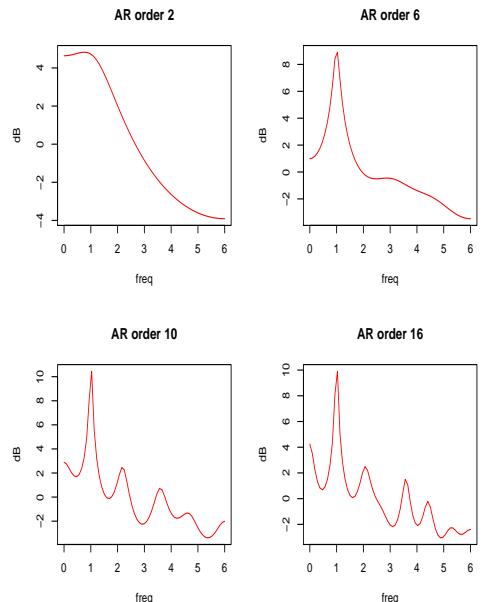
- FFT schnell für $N = 2^n$
- fft:** D -dimensionale FT, $D \in \mathbb{N}$
- mvfft:** $M \times$ eindimensionale FT

R-Programmcode

```
sp <- seq_len(N<-128) / (SF<-8000)
hertz <- c (440,883,1984)
x <- sapply (hertz, function(h)
             cos (2*pi*sp*h)) %*% (1:i:3)
z <- fft (x)
plot (x, col="blue", type="b")
plot (10*log10(Mod(z)), col="red", type="h")
rug (hertz/SF*N, tick=.1, lwd=3)
plot (180*Arg(z)/pi, col="green", type="h")
```

Anpassung eines autoregressiven Modells AR(p)

stats:::ar (x, aic=TRUE, order.max=NULL, method='yw', ...)



Funktionsargumente

x Vektor oder Zeitreihe(n)
aic Ordnung via AIC wählen?
order $\min(N - 1, 10 \cdot \log_{10}(N))$
method $\begin{cases} \text{yule-walker} \\ \text{burg, ols, mle} \end{cases}$
return Objekt der Klasse ar

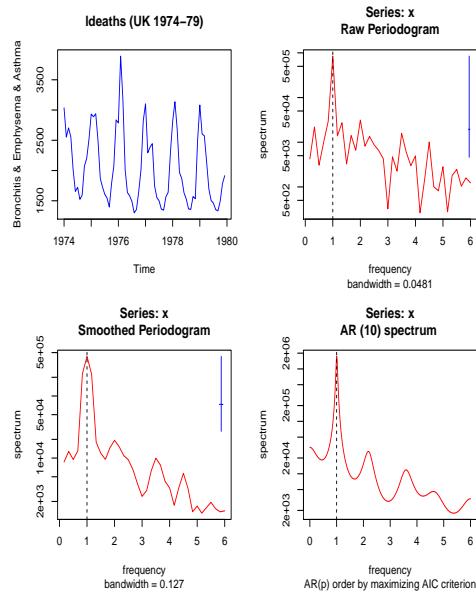
AR(p)-Modell

$$x_t - \mu =: \tilde{x}_t = e_t + \sum_{j=1}^p a_j \cdot \tilde{x}_{t-j}$$

R-Programmcode

```
M <- 128
freq <- seq (0, frequency(ldeaths)/2, len=1+M/2)
sapply (c(2,6,10,16), function(p) {
  a <- c (1, rep(0,M-1))
  a[1:p] <- -ar(ldeaths,aic=FALSE,order=p)$ar
  db <- 10*log10(1/Mod(fft(a)))[1+0:(M/2)]
  plot (freq, db, type="l", col="red")
})
```

Berechnung und Darstellung der Spektraldichte

stats:::spectrum (x, ..., method=c('pgram','ar')) \Rightarrow spec.pgram & spec.ar

Funktionsargumente

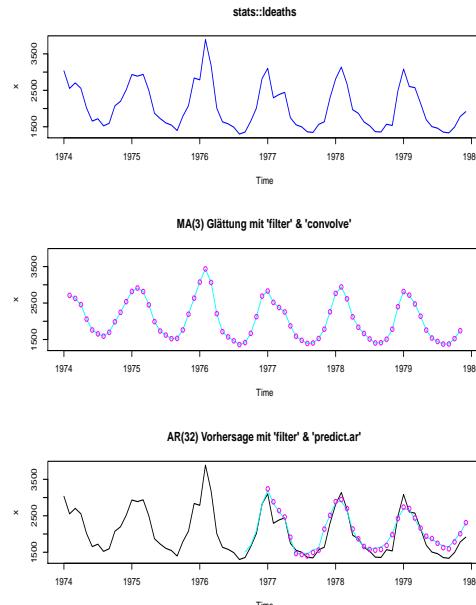
x Vektor oder Zeitreihe(n)
method Periodogramm/AR-Modell
taper=0.1 Ausblendung (cos)
pad=0 Nullanreicherung (prop.)
demean=FALSE $\hat{\mu}$ subtrahieren?
detrend=TRUE $\hat{\mu} + \hat{a}t$ subtr.?
spans=NULL Daniell-Glättung
order=NULL AR-Ordnung/AIC
freq Frequenzstützstellen
spec Spektraldichten dazu
coh,phase Kohärenz/Phase mts

R-Programmcode

```
plot (ldeaths, col="blue", ylab=NULL)
spectrum (ldeaths, col="red")
spectrum (ldeaths, spans=3, col="red")
spectrum (ldeaths, method="ar", col="red")
```

Filterung für Systeme MA(q) und AR(p)

filter (x, filter, method='convolution', sides=2, circular=FALSE, init)



Funktionsargumente

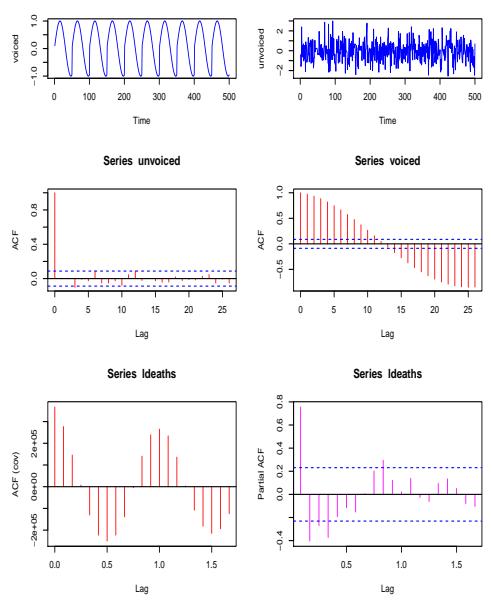
x Vektor oder Zeitreihe(n)
filter Koeffizienten (rückwärts)
method convolution/recursive
sides 1=causal, 2=center; MA
circular periodisch forts.; MA
init Burn-in (rückwärts); AR
return Zeitreihe(n)

R-Programmcode

```
plot (x, col="blue")
g <- c(1,2,1)/4
lines (filter (x, g, "c"), col="cyan")
points (time(x), c (NA, convolve (
  x, g, type="filter"), NA), col="magenta")
p <- 32
a <- ar (x, aic=FALSE, order=p)
x.1 <- window (x, end=c(1976,12))
x.2 <- predict (a, x.1, 3*p)$pred
e <- window (x, start=c(1974,p+1))
e[] <- rnorm (length(x)-p)
y <- filter (e, a$ar, "r", init=x[p:1]-mean(x))
lines (y-mean(y), col="cyan")
points (x.2, col="magenta")
```

Autokorrelationsfunktion

```
stats::acf (x, lag.max=NULL, type='corr', plot=TRUE, demean=TRUE, ...)
```



Funktionsargumente

x Vektor/Matrix oder Zeitreihe(n)
lag.max $\min(N - 1, 10 \cdot \log_{10}(N/M))$
type correlation/covariance/partial
... für den plot-Aufruf
return Objekt der Klasse acf

Autokorrelation (unnormiert)

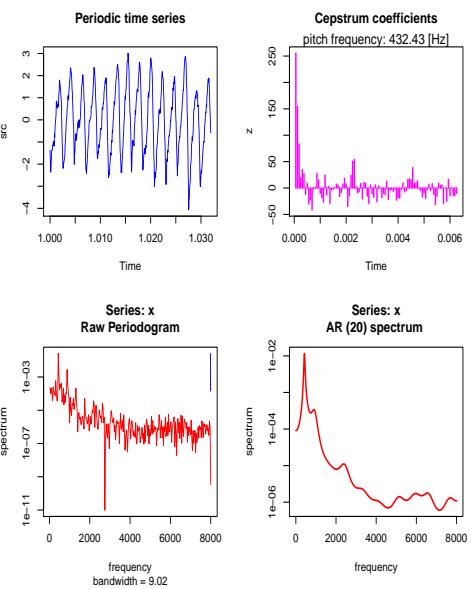
$$r_\ell \stackrel{\text{def}}{=} \frac{1}{n-m} \cdot \sum_{j=m}^{n-1} x_j \cdot x_{j-\ell}$$

R-Programmcode

```
voiced <- rep (sin (1:50/10), 10)
unvoiced <- rnorm (500)
plot.ts (voiced, col="blue")
plot.ts (unvoiced, col="blue")
acf (voiced, col="red")
acf (unvoiced, col="red")
acf (voiced, col="red")
acf (ldeaths, lag=20, "covar", col="red")
pacf (ldeaths, lag=20, col="magenta")
```

Beispiel „Grundfrequenzdetektion“

```
myutils::cepstrum (x, size=length(x)/2, k.att=0, do.pitch=0.1)
```



Funktionsargumente

x Zeitreihenobjekt (ts)
size Anzahl Ceptrumkoefizienten
k.att harmonische Verstärkung
do.pitch Strahlkonstante für F_0
return Objekt der Klasse ts

R-Programmcode

```
SF <- 16000; N <- 512; f0 <- 440
src <- (1:N)%(SF/f0)/(SF/f0)
src <- ts (src - rnorm (
  src, .5, sd=.2), freq=SF)
src <- filter (src,
  sort(runif(11)), circ=TRUE)
z <- cepstrum (src, size=100)
pfr <- attr (z, "pitch")$Hertz
pfr <- pfr[80<pfr&pfr<600]
mtext (paste (
  "pitch frequency:", 
  round (pfr[1], 2), "[Hz]"))
```

Digitale Signalverarbeitung

Merkmaltransformation und Dimensionsreduktion

Numerische und ordinale Vorhersage

Klassifikation und überwachtes Lernen

Clusteranalyse und unüberwachtes Lernen

Transformation numerischer Merkmalräume

$$\mathbf{x} = (\xi_1, \dots, \xi_N)^\top \mapsto \mathbf{y} = (\eta_1, \dots, \eta_M)^\top$$

GEGEBEN:

Ein N -dimensionaler Datensatz
 $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^N$ von
Merkmalvektoren für die Objekte
 $o_1, \dots, o_T \in \Omega$

GESUCHT:

Ein Datensatz M -dimensionaler
Merkmalvektoren $\mathbf{y}_1, \dots, \mathbf{y}_T \in \mathbb{R}^M$

GÜTEKRITERIUM:

Kleinstmöglicher Informationsverlust
der \mathbf{y}_t gegenüber \mathbf{x}_t (trotz
Dimensionsreduktion $M \ll N$)

Data Mining / EDA

Visualisierung H.D. Daten
($M \in \{1, 2, 3\}$)

Musteranalyse

Unterdrückung/Hervorhebung
störender/nützlicher Anteile

Explizite Abbildung

Finde Berechnungsvorschrift
 $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$

Implizite Abbildung

Finde Repräsentantenfolge
 $\mathbf{y}_1, \dots, \mathbf{y}_T \in \mathbb{R}^M$

Transformationsverfahren

kardinal/relational · explizit/implizit

Numerische Daten

$x_1, \dots, x_T \in \mathbb{R}^N$

linear

PCA, FA, ICA

konvex

NMF

nichtlinear

SOM, AANN

nichtparametrisch

PSA (Hauptflächen)

diskriminativ

LDA, PLS

Relationale Daten

- gemischte Skalen
- Abstand/Ähnlichkeit
- Adjazenz/Nachbarschaft

metrisch

MDS (Sammon)

Skalarprodukt

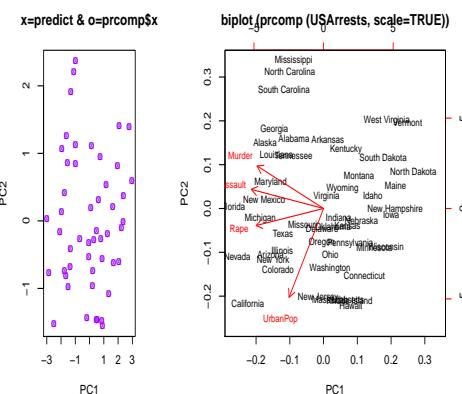
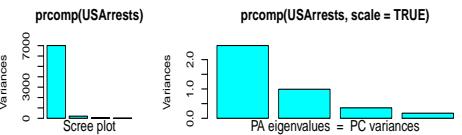
Kernel-PCA

topologisch

geodätische PCA/MDS

Principal Component Analysis (via 'svd')

`stats::prcomp(x, retx=TRUE, center=TRUE, scale.=FALSE, tol=NULL, ...)`



Funktionsargumente

- x** Datensatz: matrix, data.frame
- retx** liefere die PC von x
- center** Daten zentrieren ($\mu_n = 0$)
- scale.** Daten skalieren ($\sigma_n = 1$)
- tol** Abrissparameter für $\tilde{\sigma}_n/\tilde{\sigma}_{\max}$
- return** Objekt der Klasse prcomp

Hauptachsen/komponenten

$$X = V \cdot D \cdot U^T \quad \begin{cases} PA: U \\ PC: V \cdot D \end{cases}$$

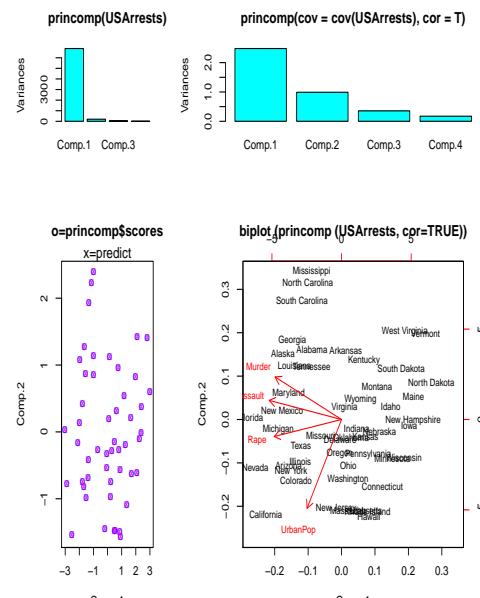
R-Programmcode

```
plot (prcomp (USArrests), col="cyan")
plot (prcomp (USArrests, scale=TRUE), col="cyan")
biplot (prcomp (USArrests, scale=T), color="blue")

o <- prcomp (formula=~., data=USArrests, scale=T)
plot (o$x[,1:2], pch=o, col="blue")
pc <- predict (o, USArrests)
points (pc[,1:2], pch=x, col="magenta")
```

Principal Component Analysis (via 'eigen')

`stats::princomp (x, cor=FALSE, scores=TRUE, covmat=NULL, ...)`



Funktionsargumente

- x** Datensatz: matrix, data.frame
- cor** correlation/covariance ?
- scores** liefere die PC von x
- covmat** Matrix, z.B. $\text{cov}.*(\mathbf{x})$
- return** Objekt d. Klasse princomp

Hauptachsen/komponenten

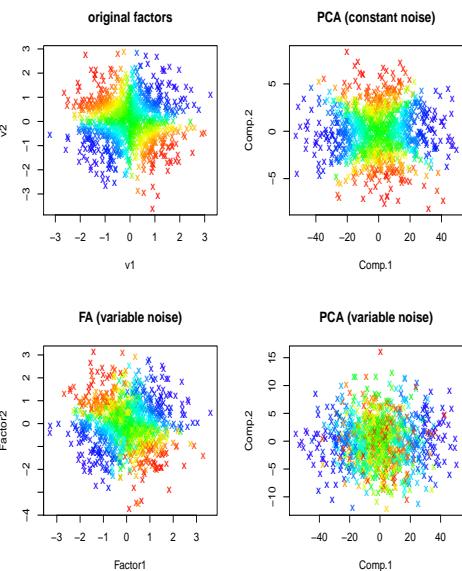
$$S = U \cdot D^2 \cdot U^T \quad \begin{cases} PA: U \\ PC: X \cdot U \end{cases}$$

R-Programmcode

```
plot (princomp (USArrests), col="cyan")
plot (princomp (cov=cov(USArrests), cor=T),
      col="cyan")
biplot (princomp (USArrests, cor=T),
        color="blue")
o <- princomp (formula=~., USArrests, cor=T)
plot (o$scores[,1:2], pch=o, col="blue")
pc <- predict (o, USArrests)
points (pc[,1:2], pch=x, col="magenta")
```

Maximum-Likelihood Faktorenanalyse

`stats::factanal (x, factors, covmat=NULL, n.obs=NA, start=NULL, scores=c('none','regression','Bartlett'), rotation='varimax')`



Funktionsargumente

- x** Matrix, Datensatz, Formel
- factors** Anzahl Ladungsvektoren
- covmat** Kovarianz (zzgl. n.obs)
- start** initiale Komunalitäten
- rotation** Fkt. Faktordrehung
- return** Klasse factanal

Faktorenanalysemodell

$$S = A \cdot A^T + R$$

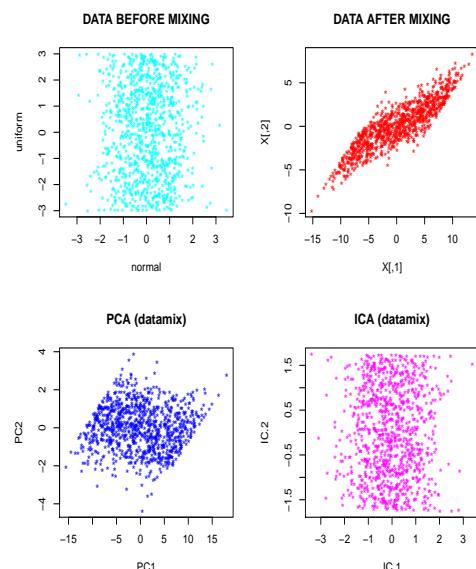
$A \in \mathbb{R}^{N \times M}$ und $R = \text{diag}(r_1, \dots, r_N)$

R-Programmcode

```
v1, v2, col=cp, main="original factors")
# X <- DATA + HOMOSCEDASTIC NOISE
plot (princomp(X)$scores[,1:2], col=cp)
# X <- DATA + HETEROSEDASTIC NOISE
plot (factanal (X, 2, scores="B")$scores, col=cp)
plot (princomp(X)$scores[,1:2], col=cp)
```

Independent Component Analysis

```
ICS:::ics (X, S1=cov, S2=cov4, S1args=list(), S2args=list(),
           stdB=c('Z','B'), stdKurt=TRUE)
```



Funktionsargumente

X Matrix oder Dataframe
 S1 Scattermatrix für 'X'
 S2 (matrix oder function)
 S12\$args Argumente zu S1, S2
 stdB Kriterium zur A^{-1} -Auswahl
 stdKurt Scatter/Formmatrizen ?
 return Objekt der Klasse ics

Quellentrennungsmodell

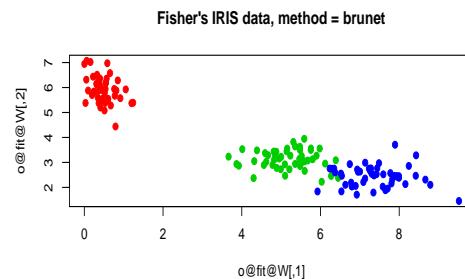
$$X = V \cdot A^\top \text{ bzw. } V = X \cdot A^{-\top}$$

R-Programmcode

```
X <- cbind (normal=rnorm(n),
            uniform=runif(n,-3,+3))
A <- rbind (c(2,3), c(2,1))
plot (X, col="cyan")
X <- X %*% t(A)
plot (X, col="red")
plot (prcomp(X)$x, col="blue")
plot (ics(X)$Scores, col="magenta")
```

Nichtnegative Matrixfaktorisierung

```
NMF:::nmf (x, rank, method="brunet", seed, ...)
```



Funktionsargumente

x Matrix oder Dataframe
 rank Anzahl Ladungsektoren
 method Iterationsschema/Zielfkt.
 seed Startkonfigurationsauswahl
 return S4-Klasse NMFfit

Konvexe Faktorisierung

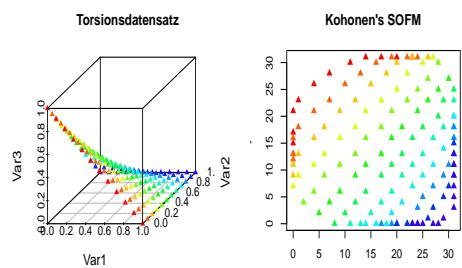
$$X \approx V \cdot A^\top \underset{\text{minim.}}{\text{minim.}} \left\{ \frac{\mathcal{D}(X||Y)}{\|X - Y\|_{\mathcal{F}}^2} \right\}$$

R-Programmcode

```
require (NMF)
for (m in c("brunet","lee")) {
  o <- nmf (iris[1:4],
             rank=2, method=m)
  plot (o@fit@W, col=1+
        unclass(iris$Species))
}
```

Kohonen's Self-Organizing Map

```
som::som (data, xdim, ydim, init='l', neigh='g', topol='r', ...)
```



Funktionsargumente

x Matrix oder Dataframe
 xdim, ydim Gitterdimensionen
 init linear/sample/random
 neigh gaussian/bubble
 topol rect/hexa
 return Objekt der Klasse som
 som\$visual Dataframe (x, y, ε)

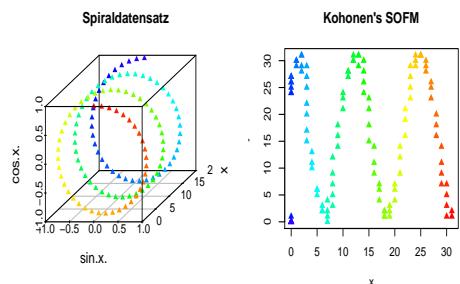
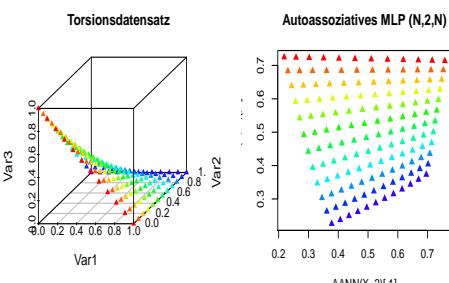
R-Programmcode

```
xy <- as.matrix (expand.grid(x,x))
X <- data.frame (xy,
                 Var3=(1-xy[,1])*(1-xy[,2]))
scatterplot3d (X, color=co)
plot (som(X,k,k)$vis[1:2], col=co)

x <- seq (0, xmax, length=n)
X <- data.frame (sin(x), x, cos(x))
scatterplot3d (X, color=co)
plot (som(X,k,k)$vis[1:2], col=co)
```

Autoassoziatives Mehrschichtenperzeptron

```
neuralnet:::neuralnet (formula, data, hidden=1, algorithm='rprop+', ...)
```



Funktionsargumente

formula Output/Input-Variablen
 data Datensatz
 hidden Zwischenschichtengrößen
 algorithm $\begin{cases} \text{backprop, sag, sl} \\ \text{rprop+, rprop-} \end{cases}$
 return Objekt der Klasse nn

R-Programmcode

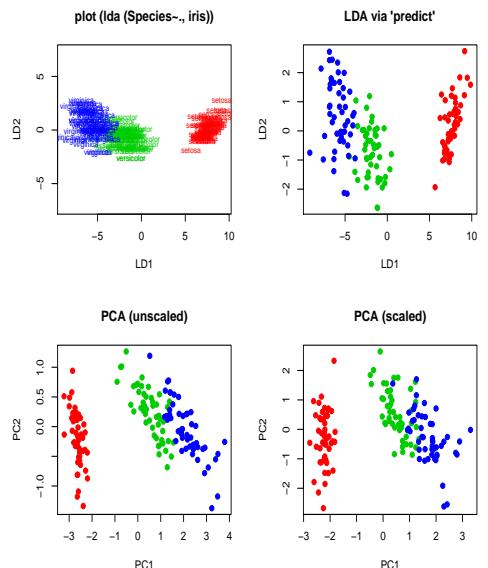
```
AANN <-function (X, k=2, sd=1/3, ...)
{
  require (neuralnet)
  X <- scale(X) * sd

  vn <- colnames(X)
  iv <- paste (vn, collapse="+")
  ov <- paste (paste (
    vn, 1, sep="."),
    collapse="+")
  fo <- formula (paste (iv, "-", ov))

  o <- neuralnet (formula=fo,
                  data.frame(X,X), hid=k, ...)
  compute(o,X)$neurons[[2]][,-1]
```

Lineare Diskriminanzanalyse

MASS::lda (x, grouping, prior=, tol=1.0e-4, method, ...)



Funktionsargumente

x Matrix oder Dataframe
grouping Klasseninfo (factor)
prior Klassenwahrscheinlichkeiten
method $\begin{cases} \text{moment} \\ \text{mle}, \text{mve}, \text{t} \end{cases}$
return Objekt der Klasse lda

Fisherdiskriminanten

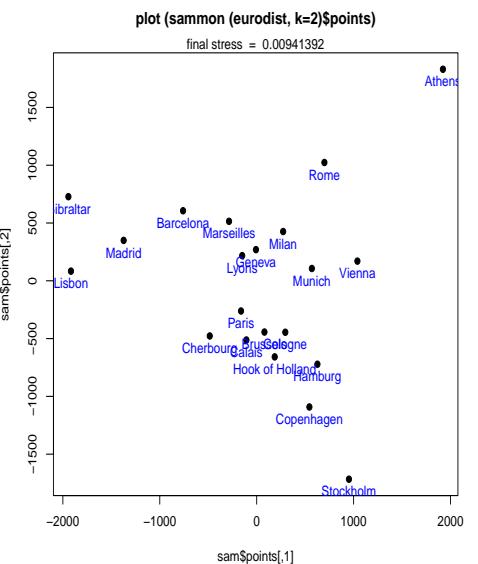
($K - 1$) Hauptträgheitsachsen von
 $Q_{\text{Kitano}} := S_W^{-1} S_B$

R-Programmcode

```
require (MASS)
ld <- lda (iris[-5], iris[[5]])
pcu <- prcomp (iris[-5], scale=F)
pcs <- prcomp (iris[-5], scale=T)
plot (ld, col=co)
plot (predict(ld)$x[,1:2], col=co)
plot (pcu$x[,1:2], col=co)
```

Mehrdimensionale Skalierung (Sammon)

MASS::sammon (d, y=cmdscale(d,k), k=2, niter=100, magic=0.2, tol=1e-4)



Funktionsargumente

d Distanzmatrix (dist, matrix)
y Startkonfiguration (k Spalten)
k Zieldimension
niter maximale Schrittzahl
points Zielkonfig. (k Spalten)
stress erzielter Schlusswert

Gütekriterium Stress

$$S_q(D, D^*) \stackrel{\text{def}}{=} \sum_{i < j} \frac{(d_{ij}^* - d_{ij})^q}{d_{ij}^q}$$

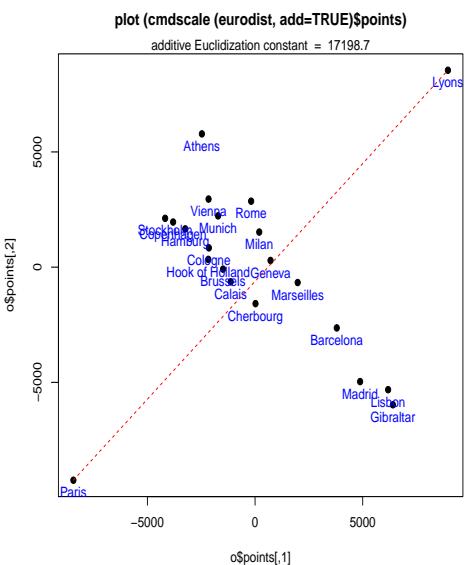
$q = 1, 2, 0$ ⚡ Sammon, relativ, absolut

R-Programmcode

```
require (MASS)
sam <- sammon (eurodist)
plot (sam$points, pch=19)
text (sam$points,
      labels=labels (eurodist),
      pos=1, col="blue")
```

Mehrdimensionale Skalierung (klassisch: duale PCA)

stats::cmdscale (d, k=2, eig=FALSE, add=FALSE, x.ret=FALSE)



Funktionsargumente

d Distanzmatrix (dist, matrix)
k Zieldimension
eig Eigenwerte abliefern?
add Euklid via c^* forcieren?
x.ret $\tilde{D} = HDH$ abliefern?
points Zielkonfig. (k Spalten)
eig, x, ac λ, \tilde{D}, c^*

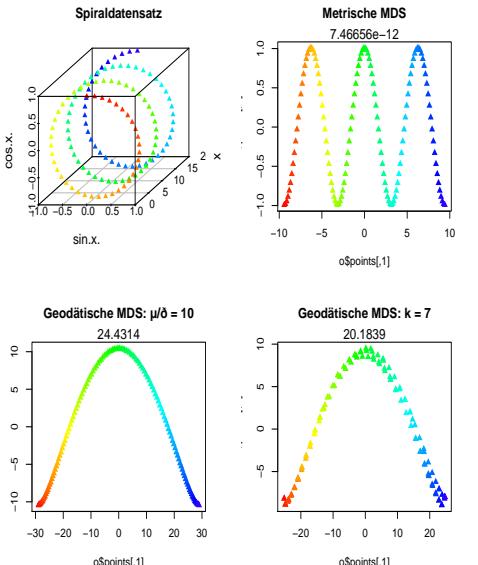
$$\text{Distanzen} \rightsquigarrow \text{Produkt} \quad \|x - y\|^2 = \|x\|^2 - 2 \cdot x^\top y + \|y\|^2$$

R-Programmcode

```
eurodist[179] <- eurodist[179]+8000 # Par
o <- cmdscale (eurodist, add=TRUE)
plot (o$points, pch=19)
text (o$points,
      labels=labels (eurodist),
      pos=1, col="blue")
lines (o$points[c("Paris", "Lyons"), ], col
```

Geodätische Skalierung

floyd_marshall (dist, makesym=c('none','first','second'), way.off=Inf)



Funktionsargumente

dist Distanzen (dist, matrix)
makesym Symmetrisierung?
way.off D_{ij} -Wert für $i \not\sim j$
return min. Pfadkosten (matrix)

Floyd-Warshall-Algorithm.

D_{ij} = minimale Distanzsumme eines Weges von i nach j

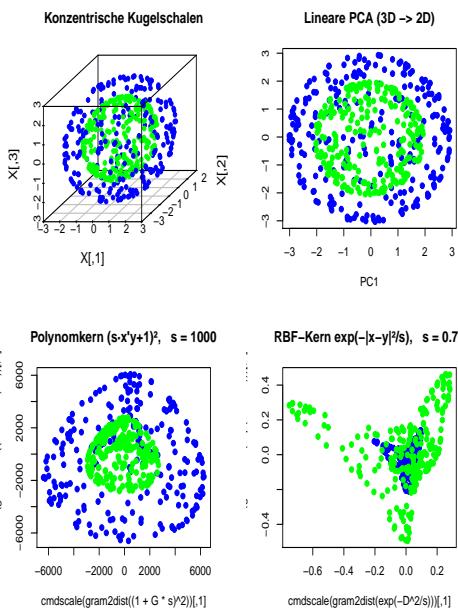
R-Programmcode

```
D <- as.matrix (dist(X))
D <- ifelse (D<mean(D)/s, 1, NA)
diag(D) <- 0
D <- floyd_marshall (D)

D <- as.matrix (dist(X))
D <- apply (D, 1, rank)
D <- ifelse (D<1+k, 1, +Inf)
D <- floyd_marshall (D, "first")
```

Nichtlineare PCA

Originaldaten \Rightarrow [Termexpansion] \Rightarrow Distanzmatrix \Rightarrow cmdscale()



Der Kernel-Trick

Originalpunkte x_1, \dots, x_T
nichtlin. Expansion $\phi x_1, \dots, \phi x_T$
 $G_{st} = \langle \phi x_s, \phi x_t \rangle = K(x_s, x_t)$
 $D_{st} = \sqrt{G_{ss} - 2 \cdot G_{st} + G_{tt}}$

\rightsquigarrow keine explizite Expansion!

R-Programmcode

```
gram2dist <- function (G) {
  D <- -2*G
  D <- sweep (D, 1, diag(G), "+")
  D <- sweep (D, 2, diag(G), "+")
  sqrt (D)
}

scatterplot3d (X, color=co)
plot (prcomp (X, scale=FALSE)$x, col=co)
G <- tcrossprod (X)
D <- as.matrix (dist (X))
plot (cmdscale (
  gram2dist ((1+G*s)^2)), col=co)
plot (cmdscale (
  gram2dist (exp(-D^2/s))), col=co)
```

Digitale Signalverarbeitung

Merkmaltransformation und Dimensionsreduktion

Numerische und ordinale Vorhersage

Klassifikation und überwachtes Lernen

Clusteranalyse und unüberwachtes Lernen

Regression ist überwachtes Lernen

Numerische Regression: $x = (\xi_1, \dots, \xi_N)^\top \mapsto y = (\eta_1, \dots, \eta_M)^\top$

GEGEBEN:

Ein N -dimensionaler Quelldatensatz $x_1, \dots, x_T \in \mathbb{R}^N$ und ein M -dimensionaler Zieldatensatz $y_1, \dots, y_T \in \mathbb{R}^M$

GESUCHT:

Eine Abbildungsvorschrift $f: \mathbb{R}^N \rightarrow \mathbb{R}^M$ mit

$$y_t \approx \hat{y}_t \stackrel{\text{def}}{=} f(x_t)$$

Gütekriterium

- Methode kleinster Quadrate
- allgemeiner Kostenansatz
- max/mean posteriori Verteilung

Dimensionalität

- | | |
|---------------|-------------|
| • univariat | $N = M = 1$ |
| • multivariat | $N > 1$ |
| • multipel | $N, M > 1$ |

Abbildungsfamilie

- linear/affin $\hat{y} = a_0 + a^\top x$
- Kerneltrick $\hat{y} = w^\top \phi(x)$
- nichtlinear $\hat{y} = f(x|\theta)$
- nichtparam. $\hat{y} = f(x | X, Y)$

Zielskalentyp

- numerisch *Ausgleichsrechnung*
- nominal *Klassifikation*
- ordinal *Präferenz*

Lineare Modelle

(linear in den Koeffizienten)

`stats::lm (formula, data, subset, weights, ..., contrasts)`

Verallgemeinert lineare Modelle

(link function: spezielle Fehlerverteilung)

`stats::glm (formula, family, data, weights, subset, ...)`

Nichtlineare Modelle

(LSE-Parameter, nichtlinearer Fkt.Prototyp)

`stats::nls (formula, data, start, control, algorithm, ...)`

Nichtparametrische Modelle

(lokale Polynomapproximation, $N \leq 4$)

`stats::loess (formula, data, ..., span=.75, degree=2, ...)`

Konnektionistische Modelle

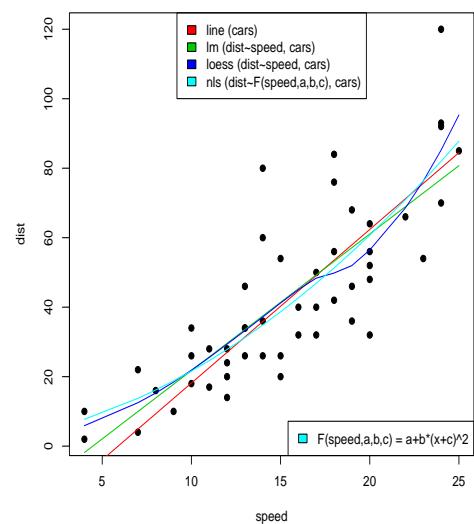
(MLP mit einer verborgenen Neuronenschicht)

`nnet::nnet (formula, data, weights, ..., size, Wts, mask, linout, entropy, softmax, censored) = F, skip=F, MaxNWts=1000, ...)`

Robuste Ausgleichsgerade (Tukey, EDA 1977)

`stats::line (x, y) (alternativ zu lm, loess, nls)`

Prädiktion 'cars\$dist' aus 'cars\$speed'



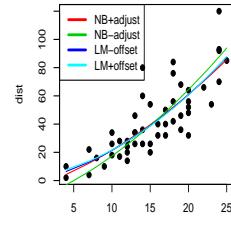
Funktionsargumente

`x, y` Quell- und Zielvektor
`return` Klasse `tukeyline`

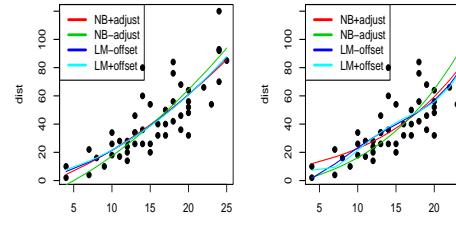
R-Programmcode

```
plot (cars, pch=19)
oln <- line (cars)
olm <- lm (dist~speed, cars)
olo <- loess (dist~speed, cars)
F <- function (x,a,b,c) a+b*(x+c)^2
ols <- nls (dist~F(speed,a,b,c),
           cars, start=list(a=1,b=1,c=1))
lines (cars$speed,
       fitted (oln), col=2)
lines (cars$speed,
       fitted (olm), col=3)
lines (cars$speed,
       fitted (olo), col=4)
lines (cars$speed,
       fitted (ols), col=5)
```

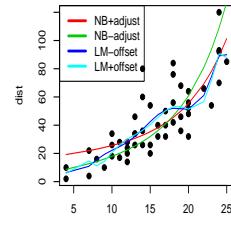
Polynomial of degree 2



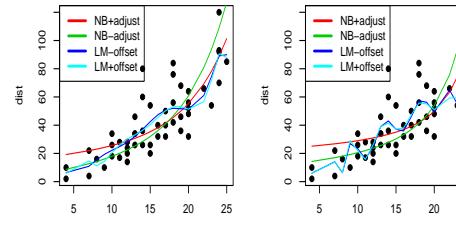
Polynomial of degree 4



Polynomial of degree 8



Polynomial of degree 16



Funktionsargumente

`x` Matrix der Quellvariablen
`y` Vektor der Zielvariable
`adjust` (lin.) Residualkompens.
`return` Objekt der Klasse `nbp`

Sternennetz-Regression

$$f(\mathbf{x}, y) = \mathcal{N}(y; \theta_0) \cdot \prod_{n=1}^N \mathcal{N}(x_n|y; \theta_n)$$

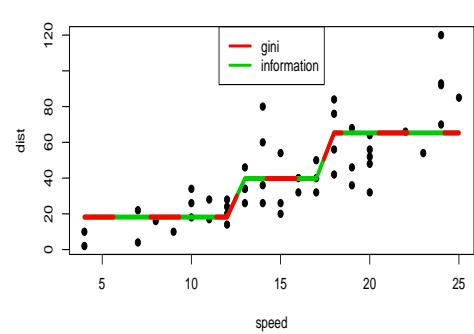
R-Programmcode

```
require (myutils)
for(p in 2^(1:4)) {
  plot (cars, pch=19)
  X <- outer (cars$speed, 1:p, "^")
  y <- predict (nbp(X,cars$dist), X)
  y <- lm.fit (X, cars$dist)$fitted
  # dto. mit 'cbind(1,X)' statt 'X'
}
```

Statistische Regressionsbäume

`rpart::rpart (formula, data, method, parms, ...)`

`cars$speed ==> cars$dist`



Funktionsargumente

`formula, data` (Formelinterface)
`method` anova,poisson,class,exp
`parms` Splittingparameter
`return` Klasse `rpart`

Entscheidungsbäume

`numeric~.` \Rightarrow `method='anova'`

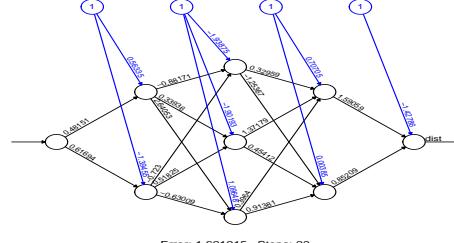
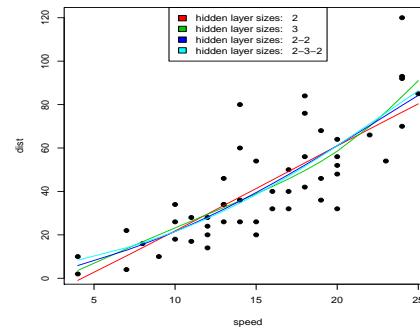
R-Programmcode

```
require (rpart)
plot (cars, pch=19)
for (s in c("gini","information")){
  o <- rpart (dist~, cars,
              parms=list(split=s))
  lines (cars$speed, predict(o), # etc.
  }
plot (o, uni=T, bran=1, comp=T)
text (o, use.n=TRUE, col="blue")
```

Mehrschichtenperzepron (MLP)

`neuralnet::neuralnet (formula, data, hidden=1, algorithm='rprop+', ...)`

MLP-Prädiktion 'cars\$dist' aus 'cars\$speed'



Funktionsargumente

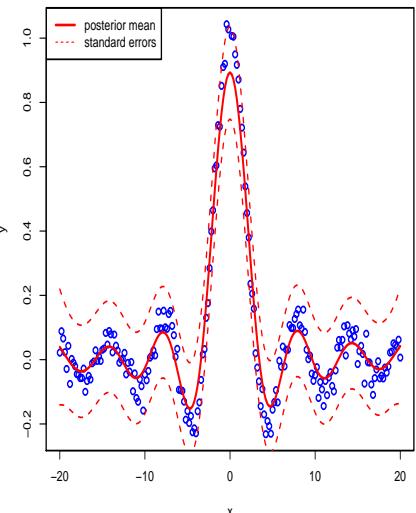
`formula` Output/Input-Variablen
`data` Datensatz
`hidden` Zwischenschichtengrößen
`algorithm` $\begin{cases} \text{backprop, sag, sl} \\ \text{rprop+, rprop-} \end{cases}$
`return` Objekt der Klasse `nn`

R-Programmcode

```
require (neuralnet)
s <- 100
H <- list (2, 3, c(2,2), c(2,3,2))
plot (cars, pch=19)
for (j in seq_along(H)) {
  o <- neuralnet (
    dist~speed, cars/s, H[[j]])
  y <- compute(o,cars$speed/s)$net
  lines (cars$speed, y*s, col=1+j)
}
plot (o, rep="best")
```

Gaußprozesse (GP)

```
kernlab::gausspr (formula, data, scaled=TRUE, kernel='rbfdot', ...)
```



Funktionsargumente

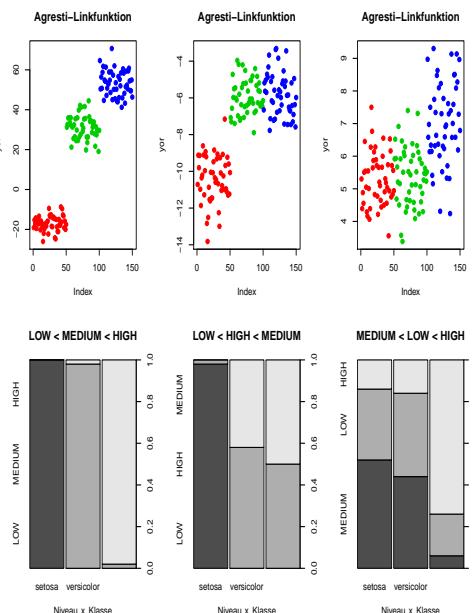
formula Output/Input-Variablen
data Datensatz
type Regression/Klassifikation
kpar Welche Kernparameter?
variance.model Error Bars?
return Objekt der Klasse gausspr

R-Programmcode

```
require (kernlab)
x <- seq(-20,20,len=n)
y <- sin(x)/x + rnorm(n, sd=0.03)
o <- gausspr (x, y,
  variance.model=TRUE)
y.m <- predict (o, x, "response")
y.s <- predict (o, x, "sdeviation")
plot (x, y, pch=21, col="blue")
lines (x, y.m, lwd=3, col="red")
lines (x, y.m+y.s, lty=2, col="red")
lines (x, y.m-y.s, lty=2, col="red")
```

Ordinal Regression (Agresti 2002)

```
MASS:::polr (formula, data, weights, start, ..., method='logistic')
```



Funktionsargumente

start Parameter $c(\text{coeff}, \text{zeta})$
method $\begin{cases} \text{logistic, probit} \\ \text{cloglog, cauchit} \end{cases}$
return Objekt der Klasse polr

Proportional-Odds LR

Gelenkfunktion für CPF: $\zeta_0, \dots, \zeta_\ell$

R-Programmcode

```
require (MASS)
for (P in list(c(1,2,3),c(1,3,2),c(2,1,3))) {
  osiris <- data.frame (iris[1:4],
    AR=ordered (Pliris$Species),
    labels=c("LOW", "MEDIUM", "HIGH")[P])
  oor <- polr (AR ~ ., osiris,
    start=c(+1,+1,-1,-1),0,1))
  yor <- as.matrix(osiris[-5]) %*% coef(oor)
  plot (yor, col=co)
  plot (iris$Species, predict(oor), ylab="") }
```

Flächenverhältnis $\frac{\text{Sepal.Length} \cdot \text{Sepal.Width}}{\text{Petal.Length} \cdot \text{Petal.Width}}$

Digitale Signalverarbeitung

Merkmaltransformation und Dimensionsreduktion

Numerische und ordinale Vorhersage

Klassifikation und überwachtes Lernen

Clusteranalyse und unüberwachtes Lernen

GEGEBEN:

Ein Quelldatensatz
 $x_1, \dots, x_T \in \Omega$ und eine
 Klassenetikettierung
 $y_1, \dots, y_T \in \{1, \dots, K\}$

GESUCHT:

Eine Entscheidungsfunktion
 $f : \Omega \rightarrow \{1, \dots, K\}$ zur
 Klassenvorhersage
 $\hat{y}_t \stackrel{\text{def}}{=} f(x_t) \stackrel{?}{=} y_t$

Gütekriterium

- Fehlerrate (Lerndaten)
- Fehlerwahrsch. (Testdaten)
- Risiko (Fehlklassif.kosten)

Bayesregel

Theoretisch optimale
 Entscheidungsvorschrift:

$$y^*(x) \stackrel{\text{def}}{=} \operatorname{argmax}_{\kappa=1..K} P(Y = \kappa | X = x)$$

Klassifikatortyp

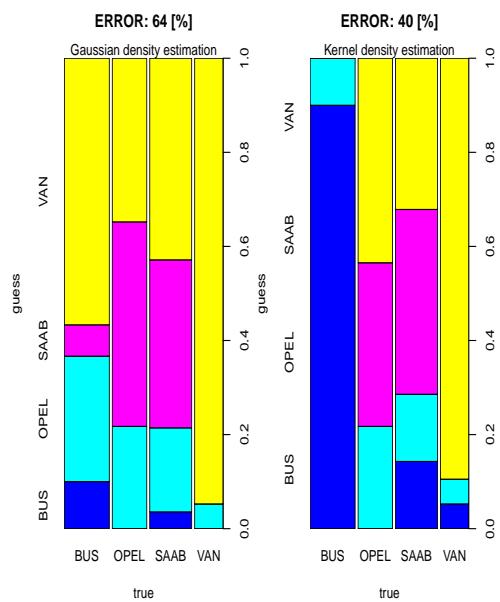
- statistisch $W.\text{modelle } f(x|\kappa)$
- diskriminativ $W.\text{modell } P(\kappa|x)$
- parametrisch $NVK, PolyK, MLP$
- nichtparametrisch $KNN\text{-Regel}$
- semiparametrisch SCT, SVM, GMK

Quellskalentyp

- numerisch
- nominal
- gemischt, metrisch, Kernel ...

Naiver Bayesklassifikator

`klaR::NaiveBayes (x, grouping, prior, usekernel=FALSE, fL=0, ...)`



Funktionsargumente

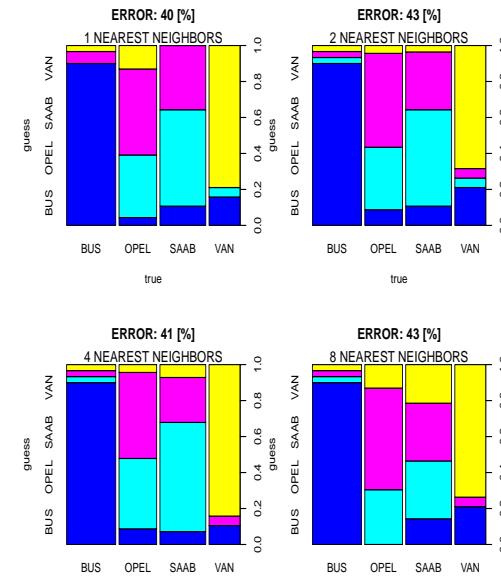
`x` Matrix oder Dataframe
`grouping` Klasseninfo (factor)
`formula, data` (Formelinterface)
`usekernel` density-Aufruf?
`fL` Faktor Laplacekorrektur
`return` Klasse NaiveBayes

R-Programmcode

```
require (klaR)
df <- lapply (load (
  "~/data/set/statlog/vehicle/rda"),
  get)
true <- df[[2]]$CLASS
o <- NaiveBayes (CLASS~, df[[1]])
guess <- predict (o, df[[2]])$class
plot (data.frame (true, guess),
  col=3+1:4)
o <- NaiveBayes (CLASS~, df[[1]],
  usekernel=TRUE)
```

k-Nächste-Nachbarn Klassifikator

`klaR::sknn (x, grouping, kn=3, gamma=0)`



Funktionsargumente

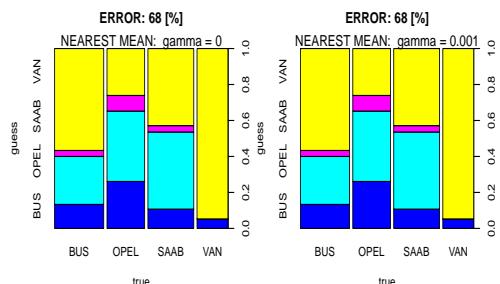
`x, grouping` Datensatz & Klasse
`formula, data` (Formelinterface)
`kn` Anzahl zu nutzender Nachbarn
`gamma` ggf. RBF-Parameter
`return` Klasse sknn

R-Programmcode

```
require (klaR)
df <- lapply (load (
  "~/data/set/statlog/vehicle/rda"),
  get)
for (k in c(1,2,4,8)) {
  o <- sknn (CLASS~, df[[1]], kn=k)
  guess <- predict (o,
    df[[2]])$class
  plot (data.frame(true,guess),
    col=3+1:4)
}
```

Minimum-Abstand Klassifikator

`klaR::nm (x, grouping, kn=3, gamma=0)`



Funktionsargumente

`x, grouping` Datensatz & Klasse
`formula, data` (Formelinterface)
`gamma` ggf. RBF-Parameter
`return` Klasse sknn

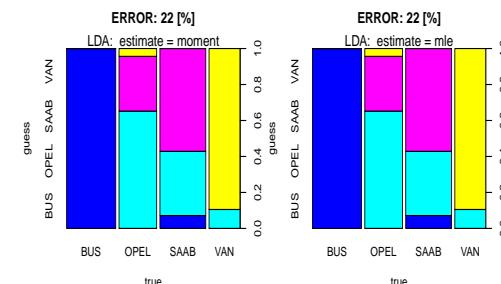
Diskriminante $(\gamma = 2\sigma^2)$
 $u_\kappa(x) = \mathcal{N}(x | \mu_\kappa, \sigma^2 \cdot E)$

R-Programmcode

```
require (klaR)
for (g in c(0,10^-c(3,6,9))) {
  o <- nm (CLASS~, df[[1]], gamma=g)
  guess <- predict (o,
    df[[2]])$class
  plot (data.frame(true,guess),
    col=3+1:4)
}
```

Mahalanobis Klassifikator

`MASS::lda (x, grouping, prior, tol=1e-4, method='moment', CV=FALSE, nu)`



Funktionsargumente

`x, grouping` Datensatz, Klasse
`formula, data` (Formelinterface)
`method` moment, mle, mve, t
`return` Klasse lda

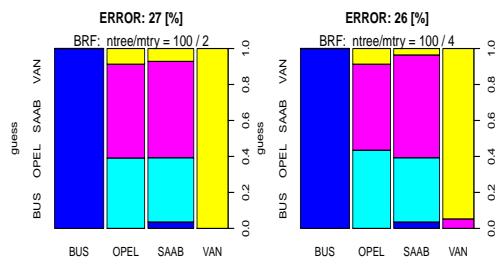
Diskriminante
 $u_\kappa(x) = p_\kappa \cdot \mathcal{N}(x | \mu_\kappa, S_0)$

R-Programmcode

```
require (MASS)
for (mth in c(
  "moment", "mle", "mve", "t")) {
  o <- lda (CLASS~, df[[1]], method=mth)
  guess <- predict (o,
    df[[2]])$class
  plot (data.frame(true,guess),
    col=3+1:4)
}
```


Breimans Zufallsensemble von Klassifikationsbäumen

`randomForest::randomForest (formula, data, ntree=500, mtry=, ...)`



Funktionsargumente

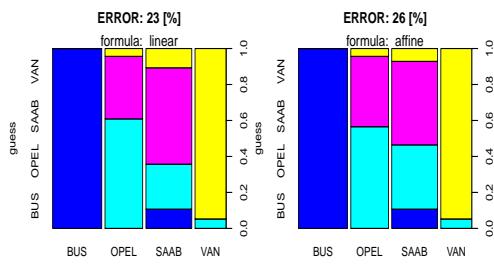
`formula, data` (*Formelinterface*)
`ntree` Anzahl Bootstrap-Bäume
`mtry` Auswahl Variablen/Frage
`replace=TRUE` Bootstraptechnik
`nodesize` Stoppkriterium (lokal)
`maxnodes` Stoppkriterium (global)
`return` Klasse `randomForest`

R-Programmcode

```
require (randomForest)
for (n in c(100,500))
  for (m in c(2,4)) {
    o <- randomForest (
      CLASS~., X[[1]],
      ntree=n, mtry=m)
    guess <- predict (o, X[[2]])
    plot (data.frame(true,guess),
          col=3+1:4)
  }
```

Linearer Quadratmittelklassifikator

`stats::lm (formula, data, singular.ok=TRUE, contrasts=NULL, offset, ...)`



Funktionsargumente

`formula, data` (*Formelinterface*)
`return` Objekt der Klasse `lm`

Lineare Diskriminante

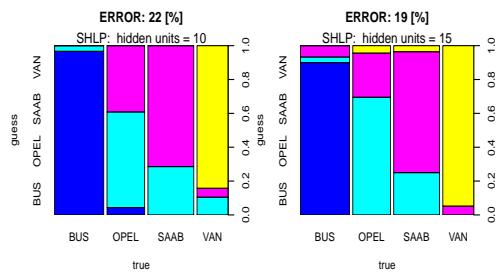
$$u_k(x) = \mathbf{a}_k^T \mathbf{x}$$

R-Programmcode

```
y <- df[[1]]$CLASS
Y <- diag(nlevels(y))[y,]
forms <- list (
  linear=Y~.-CLASS-1,
  affine=Y~.-CLASS+1, # etc.
)
for (j in seq_along(forms)) {
  o <- lm (forms[[j]], df[[1]])
  Y. <- predict (o, df[[2]])
  k. <- apply (Y., 1, which.max)
  guess <- levels(y)[k.]}
```

Dreischichtenperzeptron (SHLP)

`nnet::nnet (x, y, size, Wts, mask, linout, entropy, softmax, censored, skip=FALSE, MaxNWts=1000, ...)`



Funktionsargumente

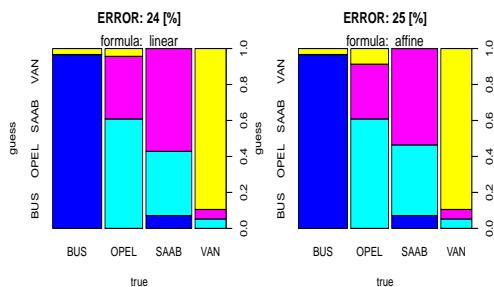
`x, y` Matrix/Dataframe für E/A
`formula, data` (*Formelinterface*)
`size` verborgene Neuronen
`skip` Kurzschlusskanten?
`Wts, MaxNWts` Startwerte, #_{max}
`return` Objekt der Klasse `nnet`

R-Programmcode

```
require (nnet)
for (h in 5*2:5) {
  o <- nnet (CLASS~., df[[1]],
             Wts=sin(1:nwts),
             MaxNWts=nwts,
             size=h)
  guess <- predict (o, df[[2]],
                  type="class")
  plot (data.frame(true,guess),
        col=3+1:4)
}
```

Log-linearer Klassifikator ($K \geq 2$ Klassen)

`nnet::multinom (formula, data, contrasts=NULL, ...)`



Funktionsargumente

`formula, data` (*Formelinterface*)
`contrasts` für nominale Attribute
`...` Argumente für `nnet()`
`return` Objekt der Klasse `nnet`

Diskriminante

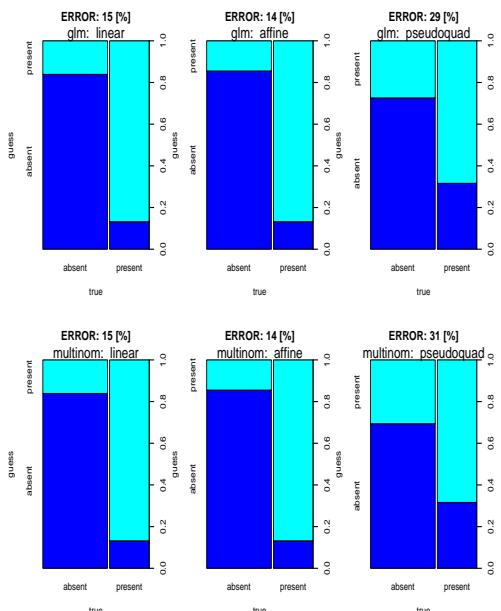
$$u_k(x) = e^{\mathbf{a}_k^T \mathbf{x}} / \sum_{\ell} e^{\mathbf{a}_{\ell}^T \mathbf{x}}$$

R-Programmcode

```
require (nnet)
fo <- list (
  linear=CLASS~.-1,
  affine=CLASS~.+1, # etc.
)
for (j in seq_along(fo)) {
  o <- multinom (fo[[j]], df[[1]])
  guess <- predict (o, df[[2]])}
```

Log-linearer Klassifikator (IRLS, $K = 2$ Klassen)

```
stats:::glm (formula, family=gaussian, data, start=NULL, ...)
```



Funktionsargumente

formula, data (*Formelinterface*)
family { gaussian, poisson, Gamma
binomial(link); quasi[...]
link { logit, probit, cauchit, [clog]log
start für IRLS (*def=OLS*)
return Objekt der Klasse *glm*

Diskriminante

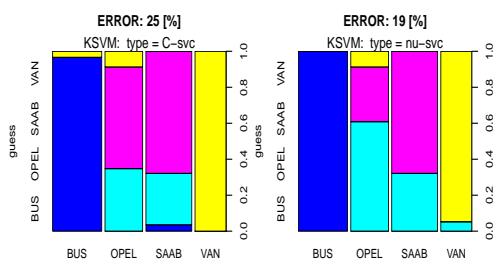
$$u(x) = e^{a^\top x} / (1 + e^{a^\top x})$$

R-Programmcode

```
# Use 'heart' dataset:  
for(j in seq_along(fo)) {  
  o <- glm (fo[[j]], df[[1]],  
            family=binomial)  
  lo <- predict (o, df[[2]])  
  guess <- levels(true) [1+(lo>0)]  
}
```

Supportvektormaschine (via Platts SMO)

```
kernlab::ksvm (x, y, scaled=TRUE, type=NULL, kernel='rbfdot',  
kpar='automatic', C=1, nu=0.2, epsilon=0.1, fit=TRUE, ...)
```



Funktionsargumente

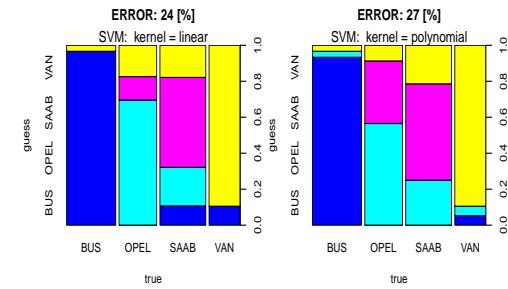
x, data (*Formelinterface*)
type { {C,nu,spoc,kbb,one}-svc
{eps,nu}-svr, eps-bsvr }
kernel { rbf,poly,vanilla,tanh }-dot
{ laplace,bessel }-dot
kpar Kernkonfiguration (*list*)
C, nu, epsilon Regularisierung
return Objekt der Klasse *svm*

R-Programmcode

```
require (kernlab)  
for (t in c("C-svc", "nu-svc", # etc.  
  o <- ksvm (CLASS~, df[[1]],  
            type=t, kernel="rbfdot")  
  guess <- predict (o, df[[2]])  
  plot (data.frame(true,guess),  
        col=3+1:4)  
}
```

Supportvektormaschine (via LIBSVM)

```
e1017::svm (x, y, scale=TRUE, type=NULL, kernel='radial', degree=3,  
gamma, coef0=0, cost=1, nu=0.5, epsilon=0.1, ...)
```



Funktionsargumente

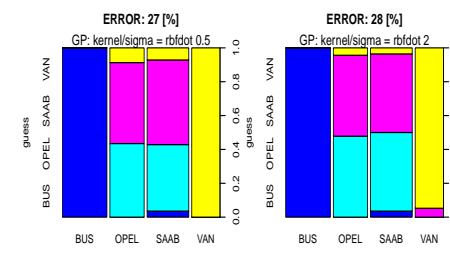
x Datenmatrix, div. Formate
y Vektor (*factor, numeric*)
type { C-class, nu-class, one-class }
{ eps-regress, nu-regress }
kernel { linear, poly, radial, sigmoid }
degree, gamma, coef0 (*Kernel*)
cost, nu, epsilon Regularisierung.
return Objekt der Klasse *svm*

R-Programmcode

```
require (e1071)  
for (k in c("linear", # etc.  
  o <- svm (CLASS~, df[[1]],  
            type="nu-class", kernel=k)  
  guess <- predict (o, df[[2]])  
  plot (data.frame(true,guess),  
        col=3+1:4)  
}
```

Gaußprozesse (GP)

```
kernlab::gausspr (formula, data, scaled=TRUE, kernel='rbfdot', ...)
```



Prädiktorargumente

object Objekt der Klasse *gausspr*
newdata Input-Variablenatz
type response, probability
coupler minpair, pkpd
return Faktor oder W'keitsmatrix

R-Programmcode

```
require (kernlab)  
for (k in c("rbfdot", "laplacedot"))  
  for (s in c(1/2,2)) {  
    o <- gausspr (CLASS~, df[[1]],  
                  kernel=k,  
                  kpa=list(sigma=s))  
    guess <- predict (o, df[[2]])  
    plot (data.frame(true,guess),  
          col=3+1:4)  
  }
```

Digitale Signalverarbeitung

Merkmaltransformation und Dimensionsreduktion

Numerische und ordinale Vorhersage

Klassifikation und überwachtes Lernen

Clusteranalyse und unüberwachtes Lernen

Die vielen Gesichter der Clusteranalyse

$$\omega = \{x_1, \dots, x_T\} \subset \Omega \quad \Rightarrow \quad \omega_1 \uplus \omega_2 \uplus \dots \uplus \omega_K = \omega$$

Gütekriterien

global:

- Likelihood (*Mischungsverteilung*)
- Verzerrung (*Gruppenprototypen*)

lokal:

- Homogenität (*divisiv*)
- Ähnlichkeit (*agglomerativ*)

Gruppenrepräsentation

$$\left\{ \begin{array}{c} \text{scharf} \\ \text{unscharf} \\ \text{gestaffelt} \end{array} \right\} \times \left\{ \begin{array}{c} \text{stetig} \\ \text{diskret} \\ \text{Metrik} \end{array} \right\} \times \left\{ \begin{array}{c} \text{sphärisch} \\ \text{geformt} \\ \text{amorph} \end{array} \right\}$$

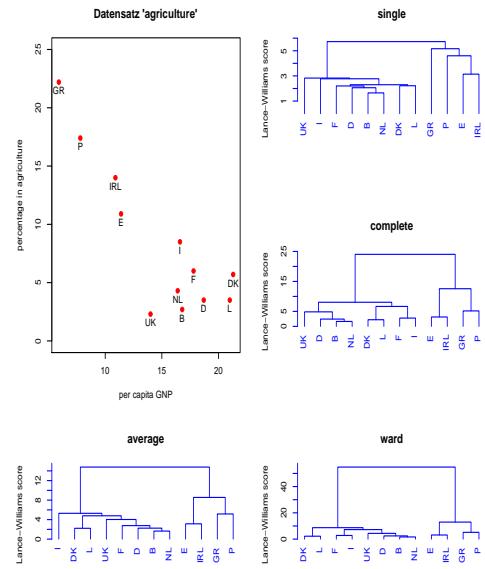
Regularisierung

Clustering $\hat{=}$ „ill-formed problem“:

- Anzahl der Gruppen
- Komplexität ihrer Gestalt

Agglomerative Gruppierung

```
stats::hclust (d, method='complete', members=NULL)
```



Funktionsargumente

d	Distanzmatrix (<i>dist</i>)
method	{ complete, single average, ward, mcquitty median, centroid }
members	(zum Neuaufsetzen)
return	Klasse <i>hclust</i>

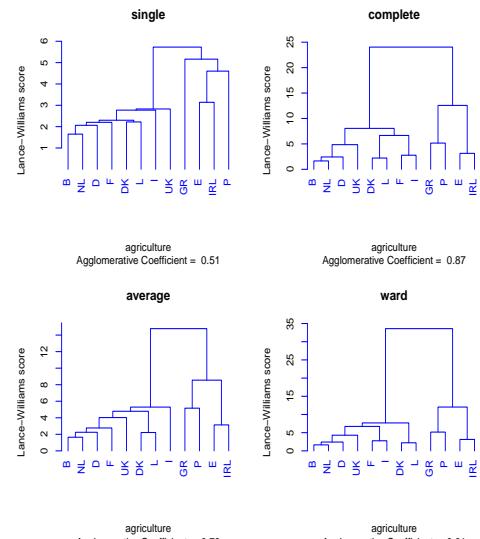
R-Programmcode

```
require (cluster) # agriculture
for(m in hcm)
  plot (hclust (
    dist (agriculture),
    method=m),
    hang=-1,
    main=m, sub="", xlab="",
    ylab="Lance-Williams score",
    col="blue")
```

⇒ *cutree* (tree, k=NULL, h=NULL)

Agglomerative Gruppierung (Lance-Williams)

```
cluster::agnes (x, diss, metric='euclidean', stand=FALSE,
  method='average', par.method)
```



Funktionsargumente

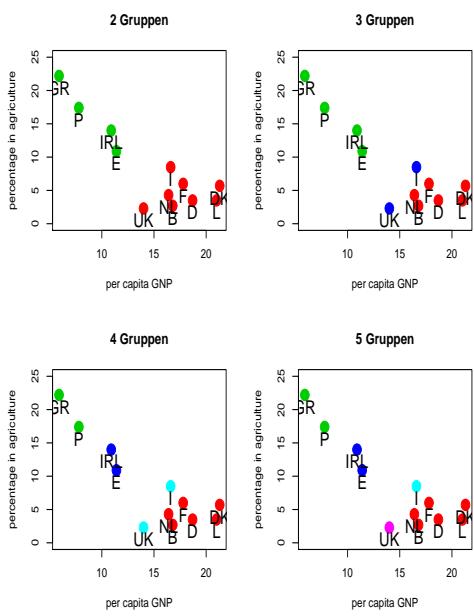
x	Distanz- oder Datenmatrix
diss	ist x Typ <i>dist</i> ?
metric	falls diss=FALSE
stand	skalare Normierung?
method	{ complete, single, average weighted, ward, flexible }
par.method	<i>LW</i> ($\alpha_1, \alpha_2, \beta, \gamma$)
return	Klasse <i>agnes</i>

R-Programmcode

```
require (cluster)
for(m in hcm) plot (
  agnes (agriculture, method=m),
  which.plots=2,
  main=m, hang=-1,
  ylab="Lance-Williams score",
  col="blue")
```

Agglomerative Gruppierung (Gaußmischung)

`mclust::hc (modelName, data, ...)` & `hclass (hcPairs, G)`



Funktionsargumente

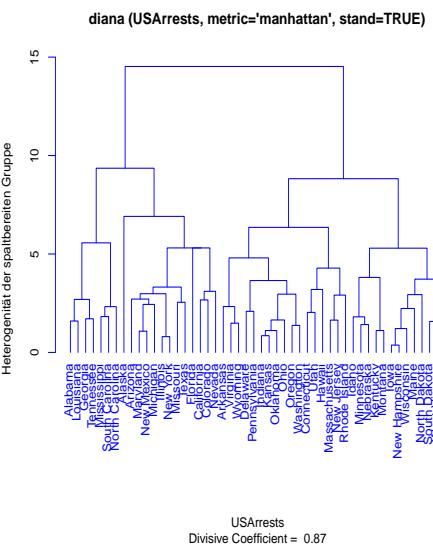
`modelName` {
E = univariat, σ_k^2 fest
V = univariat, σ_k^2 var.
EII = sphär., V_k fest
VII = sphär., V_k var.
EEE = ellipt., S_k fest
VVV = ellipt., S_k var.
}
`data` Datensatz, numerisch()
`return` (n, 2)-Indexmatrix (Aggl.)
`hcPairs` eine hc-Rückgabe
`G` welche HC-Ebenen?

R-Programmcode

```
require (mclust)
tree <- hc ("VVV", agriculture)
A <- hclass (tree, 1:5)
for (k in 2:5) {
  plot (agriculture,
    col=1+A[,k], cex=2,
    main=paste (k, "Gruppen"))
}
```

Divisive Gruppierung (Splintertechnik)

`cluster::diana (x, diss, metric='euclidean', stand=FALSE, keep.diss=n<100, keep.data=!diss)`



Funktionsargumente

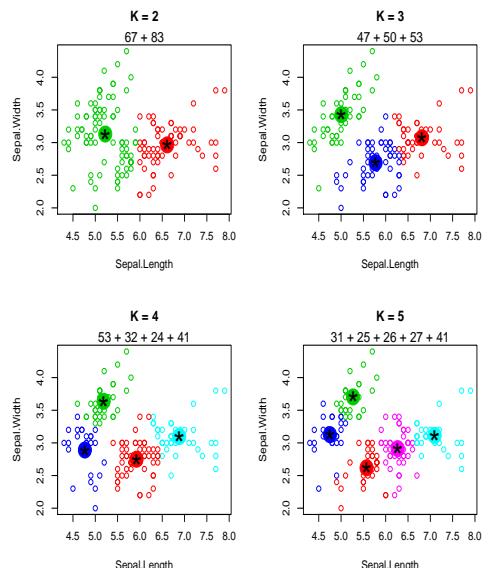
`x` Distanz- oder Datenmatrix ist x Typ dist?
`diss`
`metric` {euclidean, manhattan}
`stand` skalare Normierung?
`return` Klasse diana

R-Programmcode

```
require (cluster)
plot (diana (
  USArrests,
  metric="manhattan",
  stand=TRUE),
  hang=-1,
  col="blue",
  which.plots=2)
```

Austauschverfahren K-Means (numerisch)

`stats::kmeans (x, centers, iter.max=10, nstart=1, algorithm='Hartigan')`



Funktionsargumente

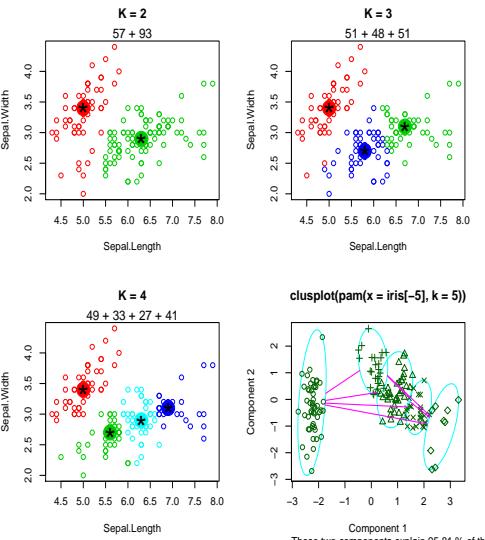
`x` Datenmatrix (numeric)
`centers` Anzahl K, Startzentren
`nstart` Anzahl Zufallsstarts
`algorithm` {Hartigan-Wong, Lloyd
Forgy, MacQueen}
`return` Klasse kmeans: {cluster
centers
size}

R-Programmcode

```
xy <- iris[,1:2]
for(k in 2:5) {
  o <- kmeans (xy, c=k, ns=10)
  plot (xy, col=1+o$cluster)
  points (o$centers,
    pch=19, cex=3, col=1+k)
  title (main=paste ("K =", k))
  mtext (paste (
    o$size, collapse=" + "))
}
```

Austauschverfahren K-Medoids (metrisch)

`cluster::pam (x, k, diss, metric='euclidean', medoids=NULL, stand=FALSE, do.swap=TRUE, trace.lev=0)`



Funktionsargumente

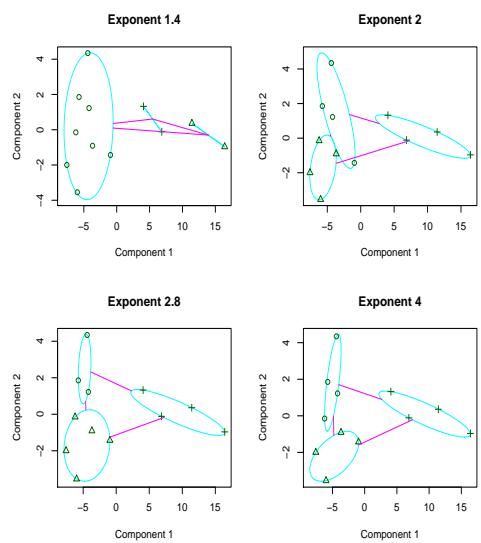
`x` Distanz- oder Datenmatrix
`k` Anzahl Gruppen
`diss` ist x Typ dist?
`metric` falls diss=FALSE
`medoids` ggf. die Startmedoide
`stand` skalare Normierung?
`do.swap` mit Schüttelpause?
`return` Klasse pam

R-Programmcode

```
require (cluster)
xy <- iris[,1:2]
for(k in 2:4) {
  o <- pam (xy, k)
  plot (xy, col=1+o$clustering)
  points (o$medoids, pch="*", cex=3)
}
clusplot (pam (iris[-5], k=5))
```

Fuzzy K-Means (metrisch)

```
cluster::fanny (x, k, diss, memb.exp=2, metric='euclidean', stand=FALSE,
iniMem.p=NULL, trace.lev=0)
```



Funktionsargumente

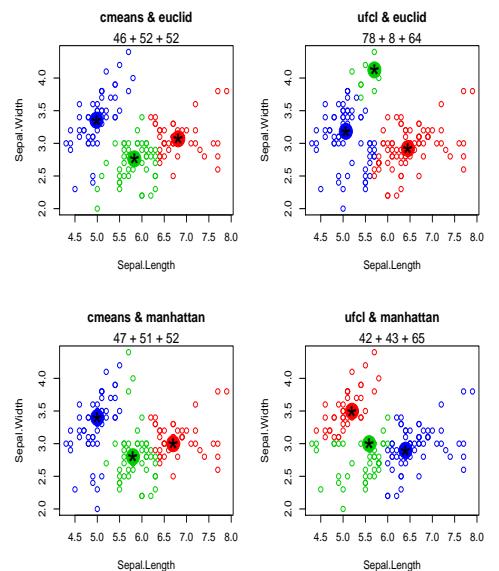
x Distanz- oder Datenmatrix
k Anzahl Gruppen
diss ist **x** Typ dist?
metric falls diss=FALSE
iniMem.p Startmemberships
stand skalare Normierung?
return Klasse fanny

R-Programmcode

```
require (cluster)
for (ex in c(1.4,2.0,2.8,4.0))
  clusplot (fanny (agriculture,
    k=3, memb.exp=ex),
    main=paste ("Exponent", ex),
    sub="")
```

Fuzzy K-Means (batch/online)

```
e1071::cmeans (x, centers, iter.max=100, dist='euclidean',
method='cmeans', m=2, rate.par=0.3)
```



Funktionsargumente

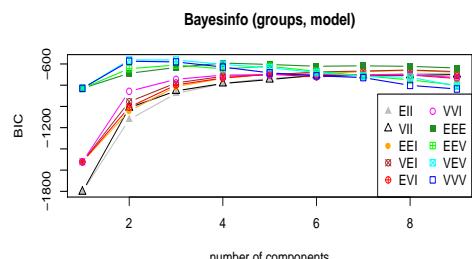
x Datenmatrix
centers #Gruppen, Startprotot.
dist {euclidean, manhattan}
method {cmeans, ufcl}
m Fuzzyexponent
rate.par Lernrate (online)
return Klasse fclust

R-Programmcode

```
require (e1071)
xy <- iris[,1:2]
for (d in c("euclid","manhattan"))
  for (mth in c("cmeans","ufcl")) {
    o <- cmeans (xy, centers=3,
      dist=d, method=mth)
    plot (xy, col=1+o$cluster)
    points (o$cent, pch="*", cex=3)
  }
```

EM-Algorithmus (Gaußmischung)

```
mclust::Mclust (data, G=1:9, modelNames=NULL, prior=NULL,
control=emControl(), initialization=NULL, warn=FALSE, ...)
```



Funktionsargumente

data Datensatz, numerisch()
G welche K -Werte? (integer)
modelName $\begin{cases} \text{Ell} = \text{sphär., } V_k \text{ fest} \\ \text{VII} = \text{sphär., } V_k \text{ var.} \\ \text{EVI} = \text{diag., } D_k \text{ var.} \\ \dots \dots \\ \text{EEE} = \text{ellipt., } S_k \text{ fest} \\ \text{VVV} = \text{ellipt., } S_k \text{ var.} \end{cases}$
prior konjugierte Fam.par.
control für EM-Aufrufe
return BIC-optimales Modell

R-Programmcode

```
require (mclust)
o <- Mclust (iris[1:4])
plot.mclustBIC (o$BIC)
o <- Mclust (iris[1:4], G=3:5)
xy <- prcomp (iris[1:4])$x[,1:2]
plot (o, data=xy, what="classification")
```

Zusammenfassung (5)

- Für die meisten Werkzeuge reserviert das 'R'-Implementierungsschema eine **Klasse** (S3/S4), einen gleichnamigen **Konstruktor** für die Lernphase und eine Methode **predict()** für die Abrupphase.
- Zur Digitalen Signalverarbeitung gibt es Methoden für das **Falten** und **Filtern**, die **Spektralanalyse**, die **Autokorrelationsanalyse** und die **Anpassung autoregressiver- und ARMA-Modelle**.
- Zur Koordinatentransformation gibt es lineare Methoden (**PCA**, **Faktorenanalyse** und **ICA**), konvexe Abbildungen (**NMF**), konnektionistische Modelle (**SOFM** und **AANN**) sowie metrik- und skalarproduktorientierte Ansätze (**MDS**, **Kernel-PCA** und **geodätische Projektionen**).
- Zur Vorhersage numerischer Attribute gibt es **LSE-** und **ML-Ansätze** mit **linearen**, **baumförmigen** oder **neuronalen** Funktionsprototypen sowie **Gaußprozesse**.
- Zur Klassifikation kennen wir **distanzbezogene** (**K-NN**), **statistische** (**NVK**), **baumförmige** (**SCT**) und zahlreiche (lineare und nichtlineare) **diskriminative** (**QMK**, **MLP**, **Logit**, **SVM**) Verfahren.
- Zur Clusteranalyse unterscheiden wir **hierarchische** (**agglomerativ** und **divisiv**), austauschende ((**fuzzy**) **K-means** und **EM-Entmischung**) und globale Partitionierter (**spektral**).