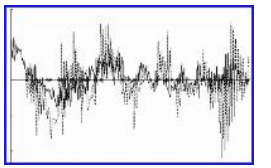
UNIVERSITÄT BAYREUTH

Lehrstuhl
Mathematik VII



R/S-Plus für Einsteiger und für Fortgeschrittene

ein Kurs über zwei Semester

entstanden in den Semestern SS 2002, WS 2002/03, SS 2003

revidiert im WS 2004/05 sowie in SS 2006 + 2007

Peter Ruckdeschel
Matthias Kohl

E-mail: peter.ruckdeschel@uni-bayreuth.de

matthias.kohl@stamats.de

Inhaltsverzeichnis

Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

. .

0 Vorwort

0.1 zur Veranstaltung

0.1.1 Rahmen

- vorliegende Folien sind Grundlage eines zweisemestrigen Kurses an der Universität Bayreuth (UBT)
- Umfang des Kurses:
  - 2 SWS für den ersten Teil (bis Kapitel 5), davon 14-tägig ca. 1 Stunde Übung
  - 2 SWS für den zweiten Teil (ab Kapitel 6), dazu 14-tägig ca. 2 Stunden Übung

- der Kurs wurde konzipiert von Dr. Peter Ruckdeschel, [peter.ruckdeschel@uni-bayreuth.de]
- die Übungen wurden konzipiert von Dr. Matthias Kohl, [matthias.kohl@www.stamats.de]
- Übungsschein: kann bei erfolgreicher Teilnahme an den Übungen oder bei Übernahme eines Referats vergeben werden
- zu Übungsaufgaben:
  - 14–tägig 4 Aufgaben
  - Abgabe per E-Mail an Dozenten
  - Vorrechnen / Präsentation der Lösung am Rechner / Beamer
  - Datensätze im WWW verfügbar

0.1.2 Adressaten / Ziele

- der Kurs wendet sich vorrangig an Studenten der Mathematik, Wirtschaftsmathematik und Technomathematik
- kann aber auch für Nicht–Mathematiker mit Interesse an Programmierung und Statistik verwendet werden
- Vorkenntnisse:
  - für Mathematiker: Stochastik I
    der Kurs wird in Bayreuth (BT) oft auch parallel zur Stochastik I angeboten und ergänzt simultan die dort präsentierten Konzepte durch ihre computertechnische Umsetzung
  - alternativ: Statistischer Methodenkurs für Nicht–Mathematiker
- der Kurs konzentriert sich auf die Umsetzung der Begriffe und Verfahren aus der Statistik in S, die Begriffe und Verfahren werden als *bekannt* vorausgesetzt

0.1.3 Anordnung des Stoffs

- der Anfang verläuft parallel zu Venables and Ripley (1999):
  - eine Kurzvorstellung von R/S-Plus in Kapitel 0
  - eine Einführung in die Sprache S in Kapitel 1
- nach diesen beiden Kapiteln stehen in der parallel dazu in BT laufenden Stochastik I zur Verfügung
  - Grundbegriffe der W–theorie
  - insbesondere der Erwartungswert
  - Gesetz der großen Zahlen (im einfachsten Fall)
  - Zentraler Grenzwertsatz (im einfachsten Fall)
- daher ziehen wir in unserem Kapitel 2 Kapitel 5 aus Venables and Ripley (1999) vor, und zwar mit den Zielen:
  - $\rightsquigarrow$ zeitnahe Umsetzung der Begriffe auf den Computer
  - $\rightsquigarrow$ frühzeitige, motivierende Simulationsbeispiele
  - $\rightsquigarrow$ Entwicklung eines intuitiven Verständnisses vom Inhalt der Grenzwertsätze

- die Tests aus Kapitel 5 von Venables and Ripley (1999) sind ausgegliedert, da diese in BT in der Stochastik I erst später zur Verfügung stehen
- unser Kapitel 2 ist erweitert um
  - einen Exkurs zur Erzeugung von Zufallszahlen; dieser ist aber optional, da die entsprechenden Verfahren in R sowieso schon zur Verfügung stehen
  - einen Abschnitt zur Simulation von Zufallszahlen und Prinzipien der Varianzreduktion, um den Studenten eine Idee von den Möglichkeiten des Rechners in diesem Bereich zu geben
- um unseren etwas mathematisch/informatisch orientierteren Hörerkreis zügig an die Programmierung heranzuführen, folgen unmittelbar die Grundzüge der Programmierung in S — basierend auf Kapitel 4 aus Venables and Ripley (1999)
- es folgt ein Kapitel zur Graphikprogrammierung basierend auf

Kapitel 3 aus Venables and Ripley (1999), erweitert um einige nützliche Tabellen

- zum Abschluss des WS / Beginn des SS stehen in der Stochastik I/II in BT die klassische Schätz– und Testtheorie zur Verfügung
  - ⇝ Schätz– und Testtheorie wird parallel dazu in Kapitel 5 aufgegriffen und in ihrer Umsetzung in R betrachtet; dazu werden
    - ⇝ Tests aus Kapitel 5 von Venables and Ripley (1999) vorgestellt
    - ⇝ Prinzipien der ML–Schätzung und die entstehenden Optimierungs– /Nullstellenprobleme besprochen
  - BT ist bekannter Standort der Robusten Statistik mit Namen wie Huber, Rieder
    - ⇝ intuitiver Zugang zu den Fragestellungen der Robusten Statistik wird in Abschnitt 5.2.3 geboten

- Teil der BT'er Hörerschaft der Stochastik I/II hat bereits Numerik I/II gehört; um einen für unsere Zwecke hinreichend einheitlichen Kenntnisstand zu erreichen:
  - führen wir in Kapitel 6 einige Grundfragestellungen der Numerik / Optimierung ein: Integration, Interpolation, Lösen von Gleichungssystemen und Optimierung
  - nennen grundlegende Lösunsgsstrategien
  - zeigen ihre Umsetzung / ihre Umsetzbarkeit in R/`S-Plus`
- viele der für den Anwender sehr interessanten, strukturierteren Modelle und darauf aufbauenden Verfahren sind in BT Inhalt weiterführender Stochastik–Veranstaltungen:
  - lineare/generalisiert lineare Modelle
  - multivariate Analyse
  - Zeitreihenanalyse
  - räumliche Statistik
- um der Hörerschaft "Appetit" auf diese Veranstaltungen zu

machen, stellen wir in Kapitel 7 jeweils kurz typische Fragestellungen vor, die zu diesen Modellen führen und präsentieren grob ihre Umsetzung in R

- den Abschluss des Kurses bildet ein Kapitel zu Fragen der fortgeschrittenen Programmierung:
  - S als objektorientierte Sprache; Vererbungsmechanismen, S4–Klassenkonzept,...
  - Prinzipien beim Schreiben eigener Bibliotheken / Pakete
  - Schnittstellen zu anderen Programmiersprachen
  - schließlich: die Organisationsstrukur des CRAN und des R Core–Teams

0.1.4 Quellen

- der Kurs folgt in großen Teilen, sowohl was Inhalt als auch Darstellung anlangt, Venables and Ripley (1999); dieses Buch haben wir ins Deutsche übertragen und auf eine "Beamerfassung" umgearbeitet
- die Anordnung unterscheidet sich jedoch beträchtlich, siehe letzter Abschnitt
- die Übungsaufgaben aus Venables and Ripley (1999) erschienen uns für unseren mathematisch orientierten Hörerkreis nicht unbedingt geeignet; daher haben wir einen eigenen Satz an Übungsaufgaben mit Lösungsvorschlägen zur Verfügung gestellt
- die Erweiterungen von Kapitel 2 verwenden Ripley (1987)
- Kapitel 5 verwendet bei der Darstellung der klassischen Schätz– und Testtheorie ohne Nachweis den Kanon der BT'er Veranstaltung Stochastik I/II, bei der Einführung in die

Fragestellungen der robusten Statistik die Vorlesungsskripten zur Veranstaltung "Robuste Statistik" in BT

- Kapitel 6 hält sich an den Kanon der BT'er Veranstaltung Numerik I/II
- Kapitel 7 verwendet dann wieder weitgehend Venables and Ripley (1999)
- Kapitel 8 bezieht sich in großen Teilen auf Chambers (1998), sowie auf Gentleman (2002) und ein DSC–Tutorial 2003 von D. Bates und T. Lumley
- daneben sind natürlich stets, wenn auch nicht immer explizit aufgeführt, zu nennen
  - die R–Hilfe
  - die R–Manuals
  - die WWW-Seiten des R–Projekts

0.1.5 zum Gebrauch des `.pdf`–Files

- es handelt sich um die Folien, die während der Vorlesung per Beamer präsentiert werden
- erfahrungsgemäß sind das ca. 30 Folien pro 90 Minuten
- die Folien liegen in einer Schwarzweiß–Fassung zum Ausdruck und in einer bunten Fassung zur Verwendung am Bildschirm / Beamer vor
- die Studenten haben in BT die Möglichkeit, sich am Lehrstuhl einen Ausdruck zum Selbstkostenpreis erstellen zu lassen; mit diesem können sie dann die Vorlesung verfolgen und Notizen ins Skript machen
- zum Ausdruck verwende man die bei fast allen Postscript–Druckern vorhandene Möglichkeit 4 Seiten auf eine `DINA4` Seite im Querformat ausgeben zu lassen
- während der Übungen und nach dem Kurs dient dieser R-Kurs

als (deutsche) Hilfe und kann im `Acrobat Reader` neben der aktuellen R–Sitzung geöffnet bleiben, um Fragen schnell zu klären

- die Folien sind mit `pdfLaTeX` aus der `MiKTeX`–Distribution erstellt; insbesondere verwenden sie das `hyperref`–Paket von Sebastian Rahtz, so dass Querverweise im Text durch Klicken verfolgt werden können, und sofern ein InterNet Anschluss im Rechner offen steht auch externen Links gefolgt werden kann
- auf die Erstellung eines Index wurde verzichtet, weil mit der Suchfunktion im `Acrobat Reader` ab Version 6.0 dies viel allgemeiner möglich ist
- die Übungsaufgaben sind gruppiert zu jeweils ca. 3–5 Aufgaben
- am Anfang einer jeden solchen Gruppe findet sich der Verweis auf die entsprechenden Kapitel im Kurs
- im Anschluss an jede Übungsaufgabe finden sich Verweise auf Lösungsvorschläge; einige Aufgaben verwenden spezielle

Datensätze, die ebenfalls verlinkt sind

  - die Lösungsvorschläge finden sich im `.pdf`–File am Ende
  - Lösungsvorschläge und Datensätze können sowohl lokal vom File als auch aus dem Netz als `ASCII`–Files bezogen werden

- bei der Beamer–/Folienversion des Kurses verwenden wir eine "Farbkodierung"; so erscheinen:
  - Überschriften
    * Überschriften der Hauptabschnitte des Files
    * Kapitelüberschriften
    * Abschnittsüberschriften
    * Unterabschnittsüberschriften
    * "Unterunterabschnitts"–Überschriften
    * laufender Text
  - Sätze etc.
    * Sätze, Theoreme, Lemmata, etc.
    * Algorithmen
    * Beispiele

- Beweise
- Beweisskizzen

– Links
- Autoren
- Links innerhalb des Texts
- bibliographische Referenzen
- Links auf Filenamen
- Links auf URLS

– und außerdem....
- R–Code
- Motivationen/erläuternder Text

0.1.6 Danksagung

- Dank geht an unseren Chef, Prof.Dr. Helmut Rieder, der uns zu der Erstellung und Veröffentlichung dieser Vorlesungsunterlagen ermunterte
- für ihre konstruktive Kritik und guten Lösungsvorschläge zu den Übungsaufgaben bedanken wir uns bei unseren beiden Software–Praktikanten
  - *Thomas Stabla*, `statho3@web.de`
  - *Florian Camphausen*, `fcampi@gmx.de`

  mit denen wir zusammen basierend auf diesem Kurs das R–Paket "distr" erstellt haben, siehe auch Ruckdeschel et al. (2006).
- ebenfalls für ihre konstruktive Kritik sowie für ihre Referate geht ein Dank an unsere eifrigen Hörer aus dem ersten Durchgang
  - *Matthias Brandl*, `brandl.matthias@web.de`
  - *Volkmar Klatt*, `CanisMaior@web.de` (— ein Geograph (!!))
  - *Sebastian Schmidt*, `sebastian.schmidt@uni-bayreuth.de`

- bei einer Revision des Kurses im WS 2004/05 halfen uns dankenswerterweise bei der Durchsicht
  - *Thomas Stabla*, `statho3@web.de`
  - *Michael Scheuerer*, `michael.scheuerer@uni-bayreuth.de`

0.2 Kurzvorstellung R/S-Plus

c.f. Venables and Ripley (1999)

0.2.1 Einsatzgebiete von R/S-Plus

- umfassende, offene Programmierumgebung
  - Werkzeuge für Statistik und Datenanalyse
  - flexibel programmierbar, echte Programmiersprache
  - matrixorientiert; objektorientiert
  - statistische Modelle in Programmier-Ausdrücke umsetzbar
  - professionelle Grafikausgabe $\rightarrow$ auch im Managementbereich zur Entscheidungsunterstützung
  - umfangreiche Import- und Exportfunktionen

0.2.2 S-Plus und R

- Ursprung: **S** (Bell-Labs, ATT jetzt Lucent!)
- kommerzielle Version: **S-Plus**

  [aktuell Version 7.0] (basiert auf S version 4)
  - Vertreiber: Insightful Corporation, früher MathSoft Inc.
  - sehr weit verbreitet (Industrie-Standard)
  - schöne graphische Benutzeroberfläche,
    WYSIWYG – but you can get more
  - komfortable Import/Export–Facilities (Grafik; Daten)
  - Schnittstelle zu Excel, PowerPoint
  - Trellis-Plots

- Open Source-Variante: **R**
  - aktuell (April 2007): Version 2.5.0, Release Date 24.04.2006; basiert auf S version 4
  - initiiert von Ross Ihaka / Robert Gentlemen (University of Auckland, Neuseeland)
  - kommandozeilen-orientierter Interpreter von S
  - komfortabel in Verbindung mit Emacs/WinEdt
  - kleiner Kern / viele Erweiterungen (über 1000 Pakete (Stand April 2007)!)
  - analoges Vertriebs–Konzept zu TEX/ LATEX, Linux
  - gnu-Lizenz, in diesem Rahmen frei verwendbar ohne Restriktionen, alle Sourcen öffentlich zugänglich
  - compilierte Codes für die Windows–, Mac– und die Linux–Welt
  - eine große, weltweite User– und Entwicklergemeinde, weit verbreitet in der Forschung

  - z.T. wesentlich schneller als S-Plus, c.f. http://www.sciviews.org/other/benchmark.htm

- R und `S-plus` zu 95% identische Syntax
- beide besitzen exzellente Manuals (mit Literatur–Referenzen)
- beide basieren auf wohl–getesteten numerischen Routinen auf `FORTRAN` Basis (Netlib,N.A.G.)
- **Der Kurs basiert auf R.**

0.2.3 Vergleich mit anderen Paketen / Sprachen

- andere Statistik-Pakete
  - abgeschlossene, datenbankorientierte Systeme
    - SAS: verbreitet in Banken– und Versicherungswirtschaft
    - SPSS: weit verbreitet in der empirischen Sozial– und Wirtschaftsforschung
    - wegen der Nähe zur Datenbank:
      - effizienter Zugang auch zu komplexen Datensätzen
      - speziell für konkurrentiellen Zugriff:
        Ablaufsicherheit vor Programmier-Mächtigkeit
        — keine `while`, nur `for` Schleifen —
      - Datensätze als Relationen statt als arrays
  - offene Systeme
    - ISP
      - "nur" auf DOS-Basis!

        - setzte Maßstäbe: interaktive, graphische Analyse
    * XploRe
        - MD-Tech: Gruppe um Prof. Härdle, HU Berlin
        - verlinkt: Dokumentation, Beispiele Programme
        - spezielle Teach-Ware
        - auch als Java-Applet im Netz

- andere Mathematik-Pakete
  - MAPLE
    * rudimentäre Statistik-Fähigkeiten
    * Stärke: symbolisches Rechnen
  - MATLAB
    * wie S: matrixorientiert
    * gibt Statistik-Modul, dem Dozenten aber nicht bekannt

- andere Programmiersprachen
  - S-Plus / R - Programme: interpretiert, nicht kompiliert

- wie jede höhere Programmiersprache:
  R / `S-Plus` potenziell erheblich langsamer als maschinennähere Sprachen
- dafür viel effizientere Notation
- sehr gute Schnittstellen zu `C` / `FORTRAN`:
  — sowohl Aufruf von `S-Plus` / R von `C` / `FORTRAN` aus als auch Nutzung von `C` / `FORTRAN` - code in `S-Plus` / R.

0.3 Vorschläge für Referatthemen

- Vor. $\hat{=}$ Voraussetzungen
- Inh. $\hat{=}$ Inhalt
- Lit. $\hat{=}$ Literatur

0.3.1 Eingabe / Import von Daten in R / S-Plus

Vor. keine

Inh.
- Datenerhebung:
  automatisiert, von Hand, Masken, Kontrollmöglichkeiten
- Datenaufbereitung (je nach Vorwissen)
  [C, PASCAL o.ä. (Beispiele), PERL, Excel, Word, lexx]
- Datenimport in R / Splus:
  per Befehl $\leftrightarrow$ interaktiv, Befehle mit Argumenten
  Datenstrukturen, in die hinein importiert wird

Lit. R-Manuals, InterNet, evtl. bei Dozenten, evtl. Vorkenntnisse

0.3.2 Graphik: die vielen Parameter von `plot` und der `par`-Befehl

Vor. keine

Inh.
- Vorstellung des Befehls `plot` mit seinen vielen, vielen Argumenten
- `legend`, `text`, `title`,
- automatisches Erzeugen / interaktiv
- Aufbereitung/Nachbereitung (extern)
- Beschriften ($\rightarrow$ TEX-Symbole!)
- `par`-Befehl

Lit. R-Manuals, InterNet, evtl. bei Dozenten, eigene Beispiele

0.3.3 Export von Daten und Graphik

Vor. keine

Inh.
- Vorstellung verschiedener Graphikformate (mit Vor- und Nachteilen)
- Datenkomprimierungs-Formate
- Einbinden von R-Grafiken in PowerPoint LaTeX /www
- Aufbereiten von Reports zur "Tischvorlage"
- Exportbefehle (Syntax, Optionen)

Lit. R-Manuals, InterNet, evtl bei Dozenten, evtl. Erfahrung mit PowerPoint etc

0.3.4 Klassen und objektorientierte Programmierung

Vor. Programmierkenntnisse

Inh.
- Paradigmen:
  - Methoden bei den Daten
  - Vererbung
  - Kapselung
  - virtuelle Methoden
- Syntax in R / `Splus`
- Klassenstruktur von R / `Splus`

Lit. R-Manuals, InterNet, evtl. bei Dozenten, evtl. Vorkenntnisse aus `C++`, `JAVA`

0.3.5 Speicherverwaltung in R / S-Plus

Vor. Programmierkenntnisse

Inh.
- dynamische Allokation von Speicher
- Freigabe
- Diagnosemoeglichkeiten
- Erfahrungsbericht
- die Befehle `options`, `assign`, `envir=...`
- Speicherverbrauch bei Zuweisungen, Kopien, Parameterübergabe

Lit. R-Manuals, InterNet, evtl. bei Dozenten, evtl. Vorkenntnisse aus Informatik?

0.3.6 Schnittstellenprogrammierung

Vor. Programmierkenntnisse

Inh.
- C $\leftrightarrow$ R oder FORTRAN $\leftrightarrow$ C
- BATCH-Programmierung
  (am besten unter Linux: core, &, ...)
- Programmierung als DLL
- Programmierung als ODE
- Beispiele

Lit. R-Manuals, InterNet, evtl. bei Dozenten

0.3.7 Bibliotheken (libraries) und Zusatzpakete (Packages)

Vor. Programmierkenntnisse

Inh.
- Wie schreibt man eine Library?
- Wie bindet man sie ein (automatisch)?
- Kurzvorstellung einer (großen) Library, am besten eine zur Robustheit
- Erfahrungsbericht

Lit. R-Manuals, InterNet, evtl. bei Dozenten, evtl. Vorkenntnisse aus Informatik?

0.3.8 Organisation von CRAN und Veröffentlichung eigener Routinen

Vor. keine

Inh.
- Foren (Welche gibt es?)
- Organisation der Nutzergemeinde
- Diagnosemöglichkeiten
- Zuständigkeiten
- Anforderungen an eigene Routinen
- Reviewing-Process

Lit. R-Manuals, InterNet, evtl. bei Dozenten

0.4 Aufbau der Sprache

c.f. Venables and Ripley (1999), pp. 1–16

- Objekte:
  - alle Ergebnisse, Ausgaben, Eingaben sind Objekte;
  - in `S-Plus` → rekonstruierbar aus DIR .Data/stock bzw. _DATA,
  - in R: im virtuellen Verzeichnis (im Arbeitsspeicher) .GlobalEnv
  - komplexere Objekte → Klassen
- Kommentare:
  #Dies ist ein Kommentar
- Listen:
  Objekte A,b werden durch **list** (A,b) zu neuem Objekt
- Ausdrücke:
  A *#gibt "Inhalt" von Objekt A aus*

- Zuweisung:
  A←8 *#weist dem Objekt A den Inhalt 8 zu*
  oder synonym A=8
- Funktionen:
  A←**function**(a1,a2=2) {a←a1∗a2; **return**(a)}

  generiert die Funktion A mit Argumenten a1,a2; a2=2 per default; Aufruf mit A(3) ergibt $3 * 2 = 6$, A(3,3) ergibt $3 * 3 = 9$
- Hilfe:
  help(**rnorm**) *#gibt die Hilfe zur Funktion rnorm() aus*

0.5 Wo bekomme ich (online-)Hilfe?

bei Fragen zu R gehe man wie folgt vor

- Lesen der R-Manuals
  - *R: An Introduction to R*
  - *R: Writing R Extensions*
  - die Manuals der eingereichten Packages auf dem CRAN
  - weitere Manuals auf den R-Seiten im Web
- Verwenden einer Kurzübersicht (siehe auch nächster Abschnitt)
- Konsultieren der R-Hilfe mit ? bzw. **help** oder **help.search**
- lesen der FAQ, zu beziehen unter
  http://cran.r-project.org/faqs.html
- Durchsuchen der R-Webseiten, z.B. mit
  http://finzi.psych.upenn.edu/search.html
- Durchsuchen der R-Mailarchive, z.B. mit
  http://maths.newcastle.edu.au/~rking/R/

- *im Entstehen begriffen (Apr. 2006):* R-Wiki
  - Konzept: Mail von Philippe Grosjean, 18 Jan 2006
  - Ziele:
    * Erstellung einer Online Dokumentation für R mithilfe der aus der Wikipedia bekannten Infra-Struktur an Wikis;
    * kleine Schritt-für-Schritt Anleitungen sollen der Allgemeinheit zur Verfügung gestellt werden
  - Mitarbeit / Nutzung:
    * siehe auch "R Wiki - About this Wiki"
  - nicht unumstritten: siehe auch Mail von Martin Mächler, 11 Jan 2006 (Hauptpunkt: Frage der Qualtiätssicherung)

- **erst dann:** Konsultieren des Maintainers des Pakets / oder der Hilfeforen:
  - dazu zuerst: Lesen der Richtlinien unter
    http://www.r-project.org/posting-guide.html
  - `r-announce`:
    http://www.stat.math.ethz.ch/mailman/listinfo/r-announce
    Ankündigungen größerer Entwicklungen / neuer Versionen; wenige Mails/Monat.
  - `r-packages`:
    http://www.stat.math.ethz.ch/mailman/listinfo/r-packages
    Ankündigungen von neu veröffentlichen Paketen oder Versionen davon
  - `r-help`: http://www.stat.math.ethz.ch/mailman/listinfo/r-help
    die Haupt R-Mailing Liste (ca 50 Mails/Tag)
  - `r-devel`:
    http://www.stat.math.ethz.ch/mailman/listinfo/r-devel
    tiefgreifendere Programmierfragen; stark moderiert; ca 5-10 Mails/Tag

0.6 eine kommentierte Literaturliste

Dieser Kurs kann kein Buch zu R/S-Plus ersetzen, sondern dient vielmehr als Kurs-Grundlage bzw. als Online-Unterstützung. . . ; (vgl. Abschnitt 0.1.5)

Wir empfehlen folgende Bücher/Kurse zum Lesen neben diesem Kurs:

- auf deutsch
  - Dolić D. (2004): *Statistik mit R*. Oldenbourg.
    Siehe auch http://www.dolic.de/R/index.html
    eher für Nicht-Mathematiker; Schwerpunkt auf elementarer Statistik; keine multivariate Statistik
  - Ligges U. (2005): *Programmieren mit* R. Springer. Siehe auch
    http://www.statistik.uni-dortmund.de/ ligges/PmitR/
    Schwerpunkt eher auf Programmierung; keine Übungsaufgaben;
  - Sachs L. und Hedderich J. (2006): *Angewandte Statistik. Methodensammlung mit* R. Springer. eigentlich wie Titel schon sagt: Methodensammlung; aber sehr gut aufbereitet und mit entsprechender Umsetzung der Verfahren in R; wenige aber sehr illustrative Übungsaufgaben zu statistischen Fragestellungen mit Lösungen;

- Sawitzki G. (2005): Einführung in S. Zu beziehen unter http://www.statlab.uni-heidelberg.de/projects/s/s.pdf.
  knapp und gut; Schwerpunkt eher auf Datenanalyse; mit Übungsaufgaben; weniger Details zur Programmierung

- auf englisch —zu R im "ganzen"
  - Chambers, J.M. (1998): *Programming with data. A guide to the S language.* Springer. Siehe auch http://cm.bell-labs.com/stat/Sbook/index.html.
    umfassend, beschreibt das "ganze" Konzept hinter S bzw. S4; keine Übungsaufgaben
  - Dalgaard P. (2002): *Introductory Statistics with* R. Springer. Siehe auch http://www.biostat.ku.dk/ pd/ISwR.html
    umfassend, kurz und präzise; besonders gut auch für absolute Computer-Laien geeignet; keine Übungsaufgaben
  - Venables W. und Ripley B. (2000): *S Programming*. Statistics and Computing. Springer. Siehe dazu auch http://www.stats.ox.ac.uk/pub/MASS3/Sprog/
    umfassend; Schwerpunkt: fortgeschrittene Programmierung (vgl.

Kapitel 8 d. Kurses); wenige, lehrreiche Übungsaufgaben (ohne Lösung)

  - Venables W. und Ripley B. (2002): *Modern Applied Statistics with S-Plus*. Springer, 4. Aufl.; siehe auch
    `http://www.stats.ox.ac.uk/pub/MASS4/`
    umfassend; gut geeignet sowohl für (elementare) Programmierung als auch für statistische Anwendungen; große Teile dieses Kurses lehnen sich an die 3. Auflage an; Übungsaufgaben eher leicht, ohne Lösung

- auf englisch —zu R zur Anwendung in der Statistik
  - Faraway J.J. (2002): *Practical Regression and Anova using* R. Zu beziehen unter
    `http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf`.
    verständliche Einführung in Regression mit R
  - Fox J. (2002): *An* R *and* `S-Plus` *Companion to Applied Regression*. Sage Publications. Siehe auch
    `http://www.socsci.mcmaster.ca/jfox/Books/Companion/`.
    verständliche Einführung in Regression mit R —gerade für Sozialwissenschaftler (Fox ist Soziologe!)

- auf englisch —Kurzübersichten

- "R reference card" von Jonathan Baron; zu beziehen unter
  http://cran.r-project.org/doc/contrib/refcard.pdf
- "R reference card" von Tom Short; zu beziehen unter
  http://cran.r-project.org/doc/contrib/Short-refcard.pdf

0.7 eine elementare Sitzung

c.f. Venables and Ripley (1999), pp. 1–16 Link aufs File

ACHTUNG: die Befehle werden en detail alle noch später erläutert. Jetzt geht es erst einmal nur darum, die Leistungsfähigkeit von R/Splus zu demonstrieren...

```
help.start() # startet die Hilfe
#
#in R einzuladen:
#    library (MASS)
#
x<-rnorm(50) # generiert 50 uiv N(0,1)
             # verteilte Variablen
             # und schreibt sie in den Vektor x
y<-rnorm(50)
#
h<-chull(x,y)       # berechnet die konvexe Huelle
```


```
        # der Punkte z[.]=(x[.],y[.])
#
plot(x,y)              #plotet die Punkte x,y
#
polygon(x[h],y[h]) # zeichnet die Huelle ein
#
objects() # listet die R-Objekte, die z.Z.
          # im .Data- DIR liegen
#
rm(x,y,h) # loescht die Objekte x,y,h
#
x<-rnorm(1000)
y<-rnorm(1000) # generiert 1000 Paare uiv N(0,1)
               # verteilter Variablen
#
hist(c(x,y+2),25) # Histogramm einer Mischung
       # von NV-s: 25 Saeulen,
```


```
        # 25 ^= Argument "nclass"
        # und Verschiebung um 2 in der 2. Komp.
#
dd ← kde2d(x,y)
# Kontour Persp.- und 3D-Plot)
contour(dd)
persp(dd, theta=-30, phi=30, d=5)
image(dd)
#
x←seq(1,20,0.5) # erzeugt (1,1.5,2,...,19.5,20)
#besser mit Argumentbezeichnungen! (aber hier:
#Zitat aus V:R:99)
#
x       # listet den Inhalt auf
#
w← 1+ x/2  # als Gewichtvektor
y← x + w*rnorm(x)
```


```
#
dum<-data.frame(x,y,w) # erzeugt Data-Frame
        # aus x,y,w
dum # und listet den Inhalt
#
rm(x,y,w)
#
fm<- lm( y ~ x, data=dum)  # passt eine einfache
        # lineare Regression von y auf x an
summary(fm) # und listet die Zusammenfassung
#
fm1<- lm( y ~ x, data=dum, weight=1/w^2)
        # passt eine gewichtete lineare Regression
        # von y auf x mit Gewichten 1/w^2 an
summary(fm1) # und listet die Zusammenfassung
#
library(modreg) # laedt die Library modreg
```


```
#
lrf <- loess(y ~ x, data=dum)
       # passt eine lokale, glatte
       # Regression von y auf x an (mit "loess")
#
attach(dum) # macht die Attribute von dum
       # als Variablen verfuegbar
plot(x,y)   # Scatterplot von y gg  x
#
lines(spline(x,fitted(lrf)))
       # plottet die lokale Regressionslinie
abline(0,1,lty=3) # fuegt die tatsaechliche
       # Regressionsgerade ein (mit einem
       # anderen Linientyp)
abline(fm)  # fuegt die ungewichtete
       # Regressionsgerade ein
       # abline() bezieht die noetige Information
```


```
        # dazu aus dem Objekt fm
abline(fm1, lty=4) # fuegt die gewichtete
        # Regressionsgerade ein -- im Linientyp 4
#
# Ausgabe moeglich ueber Kontextmenu
#          (rechte/mittlere Maustaste)
#
plot(fitted(fm), resid(fm),
          xlab="eingepasste Werte",
          ylab="Residuen")
       # ein Standard-Diagnose-Plot
       # zum Checken der Homoskedastizitaet
       # hier: Heterosked. - sichtbar?
qqnorm(resid(fm))
qqline(resid(fm))
       # Normal-Scores-Plot zum Check der
       # NV-Annahme (Schiefe, Kurtosis,
```


```
      # Ausreisser)
detach()   # abloesen des DataFrames von
      # der Suchliste
rm(fm,fm1,lrf,dum)  # aufraeumen
#
# Der Hill-Datensatz:
# Rekorde in schottischen Bergrennen
# gegen Streckenlaenge
# und Hoehenmeter
#
data(hills)  # laden des Hill-Datensatzes
hills # listing
#
pairs(hills)
attach(hills)
#
plot(dist, time)
```


```
if(interactive())
    identify(dist, time, row.names(hills))
abline(lm(time ~ dist))
        # interaktives Labeln der Datenpunkte
#
library(lqs) # laden der library (lqs)
abline(lqs(dist, time), lty=3, col=4)
        # robuste Regression mit lts
detach()
#
# Michelson-Datensatz (1879) zur Messung
# der Lichtgeschwindigkeit
#
data(michelson)
attach(michelson)
search() # welche Daten stehen R zur Verfuegung?
plot.factor(Expt, Speed,
```


```
        main="Speed␣of␣Light␣Data",
        xlab="Experiment␣No.")
        # Vergleich der 5 Experimente anhand
        # einfacher Boxplots
fm ←    aov(Speed ~ Run + Expt)
        # ANOVA-Analyse als randomisiertes
        # Block-Design mit "runs" und "Experiment"
        # als Faktoren
summary(fm)
fm0 ←   update(fm, . ~ . - Run)
        # Modell-Fit unter Weglassung der
        # unsinnigen Faktoren "runs" und
        # Vergleich der beiden Modelle
        # mithilfe einer formalen ANOVA
anova(fm0, fm)      # die ANOVA
detach()            # aufraeumen
rm(fm, fm0)         # aufraeumen
```


1 Die Sprache S / R

1.1 Grundstrukturen in R

1.1.0 Wiederholung:

siehe Abschnitt 0.4

1.1.1 Konventionen bei der Namensvergabe

- zugelassen: erstes Zeichen `a - z`, `A - Z`,
  weitere Zeichen `a - z`, `A - Z`, `0 - 9`, `.`
- Vorsicht: reservierte Namen besser nicht überschreiben
- Unterscheidung von Groß– und Kleinschreibung

1.1.2 Sprachaufbau

neben den in Abschnitt 1.1.0 wiederholten Strukturen sind wichtig:

- alle Einheiten sind *Objekte*
- Kommando–Abschluß: ; oder Zeilenumbruch
- Bilden von Blöcken durch { ... }
- Prompt: >, falls Zeile unvollständigen Befehl enthält: +
- **print** (gibt Wert einer Variablen aus), .Last.value (gibt Wert des letzten Ausdrucks aus)
- Mehrfachzuweisungen b←a←6
- Rechtszuweisung b->a oder b_a [letztere obsolet ab R 1.8.0]
- elementare *Objekte*: *Vektoren, Funktionen, Listen*

1.2 wichtige Objekte

1.2.1 Vektoren

- vorzustellen als: verkettete Zellen von gleichem "Typ" (`mode`)
- Attribute:
  - **length**, evtl.: **names**
  - **mode** (Typ): **numeric**, **character**, **complex**, **logical**, **list**, **function**
- Zugriff auf Elemente: `a[4]`
- R–Beispiel 1.2-1:

```
a<-seq(1,6,1)   # fuellt a mit 1,..,6
                #(siehe auch Abschnitt 1.3.1)
length(a)       # Laenge von a
```


```
mode(a)          # Modus von a
a[3]             # gibt Element Nr. 3 von a aus
a[4]←3           # setzt Element Nr. 4 auf 3
names(a)←c('a','b','c','d','e','f')
 # benennt die Elemente des Vektors a als a ... f
a
names(a)
a["d"] # jetzt auch Zugriff ueber Namen moeglich
```

1.2.2 Matrizen und Arrays

Matrizen und Arrays sind Verallgemeinerungen von Vektoren mit zwei/mehreren Indizes.

- Zugriff auf Elemente: a[4,2] bzw. a[4,2,3]
- Zugriff auf alle Elemente einer/mehrerer Koordinaten: a[4,] bzw. a[,2,]
- Erzeugung mit den Funktionen **matrix**(·) bzw. **array**(·); diese haben als Argumente

- **data**: ein Vektor mit Elementen, mit denen die Matrix / das Array zu füllen ist; beachte auch Abschnitt 1.4.2
- **nrow**, **ncol**: (bei **matrix**) Zahl der Zeilen und Spalten
- **dim**: (bei **array**) ein Vektor in der Länge der Zahl der Indizes; enthält Dimensionen der einzelnen Indizes
- byrow: (bei **matrix**) logischer Wert; falls TRUE werden zuerst die Zeilen, dann die Spalten gefüllt; sonst (Voreinstellung) umgekehrte Reihenfolge; siehe

```
matrix(data=1:4,nrow=2,ncol=2) # byrow=FALSE
matrix(data=1:4,nrow=2,ncol=2,byrow=TRUE)
```

- **dimnames** optional: Namen der Dimensionen / Indexbereiche

- Erzeugung von Matrizen und Arrays aus anderen Datentypen siehe Abschnitt 1.2.7

1.2.3 Listen

- vorzustellen als: verkettete Zellen von möglicherweise verschiedenem "Typ" (`mode`)
- Bilden von Listen wie in Abschnitt 0.4:
  A←**list**(a=ae3,del=del) oder A←**c**(ae3,del) — **c** ohne **names**
- Zugriff auf Listenelemente: A[[2]] bzw. A**$**del oder A**$d**
- Auflösen einer Liste durch **unlist**(A)

R–Beispiel 1.2-2 [Vektoren, Matrizen etc.]:

Aufgabe:

(a) Erzeugen Sie einen Vektor `v` mit Wertebelegung $1, \ldots, 30$.

(b) Modifizieren Sie alle Werte kleiner als 5 auf 5 und alle größer als 20 auf 20

(c) Machen Sie aus dem Vektor eine 3×10–Matrix `M` und geben Sie die zweite Spalte aus.

(d) Machen Sie aus der Matrix ein Array `A` mit Dimensionen $2 \times 3 \times 5$ und geben sie alle Elemente mit drittem Index 3 aus.

(e) Erzeugen Sie aus `v`, `M`, `A` eine Liste mit Namen `v, M, A`. Geben Sie das Element $2, 3$ des zweiten Listenelementes aus.

```
##################################################
# Teil (a)
##################################################
v ← 1:30

# oder:
v ← seq(from=1, to=30, by=1)

# oder:
v ← seq(form=1, to=30, length=30)

# oder (umstaendlich, aufwendig):
```


```
v ← vector("numeric", length=30);
for(i in 1:30) {v[i] ← i}


##############################################################
# Teil (b)
##############################################################
v[v>=20] ← 20
v[v<=5] ← 5


##############################################################
# Teil (c)
##############################################################
M ← matrix(data=v, nrow=3, ncol=10)
M[,2]
```


```
##################################################
# Teil (d)
##################################################
A <- array(data=v, dim=c(2,3,5))
A[,,3]


##################################################
# Teil (e)
##################################################
L <- list(v=v,M=M,A=A)
rm(v,M,A)
L[[2]][2,3]
#bzw. L$M[2,3]
```


1.2.4 Funktionen I

1.2.4 (a) Funktionsdeklaration

- wie in Abschnitt 0.4 — beachte "lazy calling"
- formal: `<name>`←**function**(`<arg1>`,`<arg2>`, ...) `<expression>`

1.2.4 (b) Funktionsaufruf

- unspezifizierte Argumente
  z.B.: **c** (...) , **pmax**(...), **max**(...)
- spezifizierte Argumente
  - Argumente in der korrekten Reihenfolge
    polygon(x1, y1, F, 10)
  - Argumente spezifiziert als `<name>=wert`:
    polygon(y=y1, x=x1, **density**=10, border=F)

– gemischt: **polygon**(x1, y1, border=F, **density**=10)

1.2.5 Faktoren

c.f. Abschnitte 1.8 und 7.1.1 (c)

- spezielle String-Vektoren
- signalisieren statistischen Verfahren:
  kein Label sondern *kategorielle* Variable!
- Syntax: bg←**factor**(**c**("D","N","B","D","E","D","F","I"))
- spezieller **print**-Befehl: **print** . **default** ⇝ interne Codierung
- zus. Levels durch **factor**(**c**("a","b"), **levels** =**c**("a","b","c"))
- Ordnung def'bar (alphabet. per default): (hier t<m<h)
  ek←**ordered**(**c**("h","t","m","t")); ik←**ordered**(**c**("h","m","t"))

1.2.6 *Data-Frames*

- üblicher Typ zum Ablegen von Datenmatrizen
  $\rightsquigarrow$ mit Einträgen versehene Relation in DB–Sprechweise
- "Zeilen" vom gleichen Typ
  $\rightsquigarrow$ Beobachtungen/Messungen in stat. Sprechweise,
  $\rightsquigarrow$ Eintrag in DB–Sprechweise
- analog zu Listen:
  "Spalten" möglicherweise von unterschiedlichen Typen
  $\rightsquigarrow$ Merkmale in stat. Sprechweise,
  $\rightsquigarrow$ Attribute in DB–Sprechweise
- Operationen **cbind**: Spalten anhängen, **rbind**: Zeilen anhängen
- weitere DB–Operationen möglich

- R–Beispiel 1.2-3:

```
library(MASS)   # laedt die MASS–library
data(painters)  # laedt den Datensatz "painters"
                #(siehe auch Abschnitt 1.3.2)
painters        # gibt ihn aus
row.names(painters)    # Zeilennamen (Malernamen)
painters[1:5, c(2,5)] # spezielle Auswahl
```

1.2.7 Typ-Umwandlung / Casting

- implizites Casting: siehe Abschnitt 1.4.1
- explizites Casting: `as.xxx`,
  z.B. wandelt `as.matrix(data)` den numerischen *Data-Frame* `data` in eine Matrix
- Typ-Check: `is.xxx`

1.3 Dateneingabe

1.3.1 Eingabe von Hand

- mühsam und fehleranfällig, besser mit spezialisierter Software: z.B.

  http://www.hrz.uni-dortmund.de/A1/kurse/unterlag/statistik/kodier.html

  http://www.hrz.uni-dortmund.de/A1/kurse/unterlag/statistik/maske.html

- zur Not:
  - durch Nutzung von daten←**scan**(), abzuschließen durch <ctrl>-D
  - daten←**c** (−1,34.3,0.3,....) — **c** ≙ *concatenate*

1.3.2 Automatisches Füllen von Objekten

- der Aufzählungsdoppelpunkt 34.23:100
- der **seq**–Befehl: generiert reguläre Folgen;
  Syntax **seq**(from,to,by=) oder **seq**(from,to,length=)
- der **rep**–Befehl: z.B. x←**rep**(4,times=5)
- besonders nützlich bei Designs, siehe letzte zwei Zeilen im folgenden Beispiel
- R–Beispiel 1.3-1:

```
x← 1:4          # c(1,2,3,4)->x
i←rep(2,4)      # c(2,2,2,2)->i
y←rep(x,2)      # c(1,2,3,4,1,2,3,4)->y
z←rep(x,i)      # c(1,1,2,2,3,3,4,4)->z
w←rep(x,x)      # c(1,2,2,3,3,3,4,4,4,4)->w
##################################################
```


```
# 2way design
# 4 Zeilenklassen
# 3 Spaltenklassen
# jeweils 2 Beobachtungen
##########################################################
colc <- rep(1:3, rep(8,3))
# ergibt:
# 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
rowc <- rep(rep(1:4, rep(2,4)),3)
# ergibt:
# 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```


1.3.3 Einlesen von Daten I

Peter Ruckdeschel
Matthias Kohl

1.3.3 (a) der read.table–Befehl

- liest Data-Frame ein
- die erste Zeile des einzulesenden Files sollte einen Namen für jedes Merkmal / jede Variable enthalten
- jede weitere Zeile sollte mit einem Beobachtungs-Label (*row label*) beginnen
- oft sinnvoll den Beobachtungs-Label zu überlesen $\rightsquigarrow$ **read.table**(<filename>,header=T)

BEMERKUNG 1.3-2 [FILENAMEN]:

(i) \'s müssen in Windows maskiert werden, so dass wir z.B. erhalten `"C:\\mywork\\r\\sws\\file.dat"`; stattdessen

können wir aber auch schreiben
`"C:/mywork/r/sws/file.dat"`.

(ii) Der Filename `clipboard` kann genutzt werden um Daten aus der Zwischenablage zu lesen.

1.3.3 (b) der scan–Befehl

- liest einen einzelnen Vektor ein
- sehr flexibel, c.f. Referat
- **count.fields** zählt die Beobachtungen
- einfache mögliche weitere Parameter what, multi.line, sep, siehe Beispiel
- R–Beispiel 1.3-3:

```
M← matrix(scan("mat.dat"),ncol=4,byrow=T)
```

```
  # liest Daten in eine 4spaltige Matrix
iD<-scan("inp.dat",what=list(id="",x=0,y=0))
  # liest Daten in eine Liste aus 3 Komp.
  # erste char, andere beide numeric
iLc<-as.list(rep(0,50)) # casting in Liste
names(iL)<-paste("X",1:50,sep="")
dL<-scan("big.dat", what=iL, multi.line=T)
  # einlesen von 50 Datensaetzen
  # jeder mit 5 Zeilen im File
  # in die Variablen X1,....,X50
  ### zu "paste" siehe Abschnitt 1.6
```

1.3.3 (c) der data–Befehl

Liegen die Daten in einer Library von R, so können diese mit dem Befehl **data**(<Datenname>,**package**=<packagename>) eingeladen werden

1.3.4 Einlesen von Daten II — Dateneingabe und Import unter R / S-Plus

— basierend auf einem Referat von *Volkmar Klatt* vom 29.04.2002

1.3.4 (a) Gliederung

- Was sind Daten?
- Der Kontext von wissenschaftlicher Datenproduktion
- Wie „macht“ man Daten-Dateien?
  - Beispiel: Dateneingabe über SPSS
  - Beispiel: Dateneingabe über Access
- Was muß man über fremde Daten mindestens wissen?

- Datenaufbereitung und Import nach R
  - Beispiel: Import von SPSS-Dateien
  - Beispiel: Import einer ASCII-Datei
  - weitere interessante Importmöglichkeiten
- Datenstruktur(en) der eingelesenen Daten in R

1.3.4 (b) Was sind „Daten“?

- Daten sind kodierte Informationen über „die“ Wirklichkeit und die Erläuterung dieser Kodierung
- Daten sind kein Selbstzweck, sondern dienen einem Ziel (sollten;-))

1.3.4 (c) Der Kontext von wissenschaftlicher Datenproduktion

Die 4 Phasen einer (sozial-)wissenschaftlichen Untersuchung:
(Die **fett** gedruckten Abschnitte wollen wir hier kurz ansprechen)

1. Definitionsphase
   - Problemwahl
   - Literaturanalyse; eigene Hypothesen bilden
   - theoretischen Bezugsrahmen aufstellen

ergänzt nach von Alemann (1984), vgl. besonders Kapitel 3, “Die Phasen einer sozialwissenschaftlichen Untersuchung.”

- **Operationalisierung** der Grundbegriffe (*wie* messe ich?)
- Festlegen von Grundgesamtheit und Analyseeinheit
- Forschungsplan aufstellen, darin: wichtigste Ziele, Gliederung, Literaturverzeichnis, Hilfsmittel festlegen; Zeit und Kosten abschätzen

2. Durchführungsphase: Forschungsplan wird umgesetzt, d.h. Forschungsinstrumente konkretisiert
   - Auswahlplan erstellen: Wer wird befragt? (Auswahlverfahren)
   - **Kodierung der Daten festlegen** -⇒ maschinenlesbare Form
   - **Vortest** (=Pre-Test) und **Exploration**: Ist Forschungsinstrument praxistauglich? Datenanalyse mit Dummy-Daten testen
   - **Datei für die Auswertung erstellen, Eingabemaske erstellen**, Datenanalyse mit Dummy-Daten testen
   - Vorbereitung der Hauptuntersuchung (Geld und Zeit bereitstellen, Schulung der Hilfskräfte, Durchführung der

Hauptuntersuchung)

3. Analysephase: **Daten bereinigen**, d.h. eindeutig fehlerhafte Messwerte kennzeichnen
   - Statistikprogramm starten
   - Durchführung der Analyse (**Daten eingeben/einlesen**, Berechnungen durchführen)
   - Aufbereitung der Daten (Visualisierung)

4. Disseminationsphase:
   - Schreiben der Forschungsberichte
   - Publikation und Verbreitung der Ergebnisse / Vortrag halten
   - Geld und Lorbeeren einstecken

1.3.4 (d) Wie „macht“ man Daten–Dateien?

Beispiel 1.3-4 [Dateneingabe über SPSS]:

- Es können sehr komfortabel Level vergeben werden (und nach R importiert werden)
- Nachteil: bei großen Tabellen ist es schwierig, nicht in falsche Zeilen bzw. Spalten zu rutschen

---

Dissemination ist eigentlich ein medizinisches Fachwort und bedeutet: Ausbreitung einer Seuche ...

Beispiel 1.3-5 [Dateneingabe über Access]:

- persönlich gestaltete Eingabemasken sind möglich
- Überprüfung während der Eingabe auf Tippfehler möglich!
- Nachteil: Das Anlegen der Tabellenstruktur und Masken ist zeitaufwändig

Mit einer Validitätsprüfung der eingetippten Werte kann man sich in Access auf Unstimmigkeiten hinweisen lassen.

1.3.4 (e) Was muss man über fremde Daten mindestens wissen?

- Datenquelle - Zuverlässigkeit, Messverfahren
- rechtliche Restriktionen – darf ich die Daten verwenden?
- welches Skalenniveau haben die Daten?
- wie sind Metainformationen kodiert (wie "Antwort verweigert", trifft nicht zu", "keine Angaben")

Tipps zur Datenspeicherung:

- Daten aus mehreren Quellen getrennt halten, z.B. indem für jede Quelle eine SQL-Tabelle anlegt und diese verknüpft
- Daten in simpler Textdatei (kein Word usw.!) beschreiben, dies auch ausdrucken.

1.3.4 (f) Datenaufbereitung und Import nach R

Beispiel 1.3-6 [Import von SPSS-Dateien]:

siehe Datei

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/referate/volkmar/spss_import.R

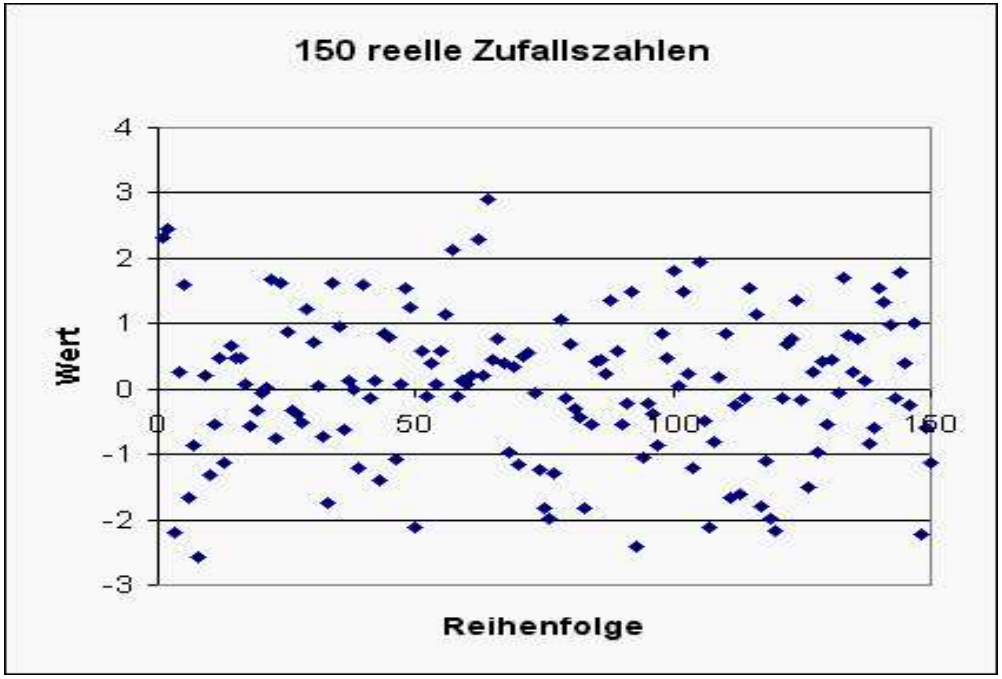
Beispiel 1.3-7 [Import einer ASCII-Datei]:

siehe Datei

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/referate/volkmar/tabellen_import.R

Beispiel 1.3-8 [weitere interessante Importmöglichkeiten]:

- scan: allgemeinere Methode als read.table()
- unter Windows: Anbindung an Excel via DCOM-Server



Unter Excel wurden über die R-Schnittstelle 150 Zufallszahlen erzeugt und mit Excel dann als Diagramm dargestellt.

- Import von Daten ist einfach

- Excel-Blätter sehen dabei jedoch schnell unübersichtlich aus
- Der DCOM-Server stürzt schnell ab, kann aber auch schnell wieder gestartet werden
- Anbindung an eine SQL-Datenbank
  - Zugriff auf extrem große Datenbestände ist möglich
  - umständlicher und störanfälliger als eine Datendatei, u.U. verminderte Datensicherheit
  - auch über's Internet möglich
  - es gibt kein einheitliches SQL-Modul für R, sondern viele verschiedene

R–Beispiel 1.3-9 [Import von SPSS-Dateien]:

```
# 2002 (c) Volkmar Klatt
# volkmar.klatt@stud.uni-bayreuth.de
#
##########################################################
# Ein Beispiel zum Einlesen von Dateien im
# SPSS-Format mit dem Statistikprogramm R
# (verwendet wurde hier R, Version 1.4.1 und
#  das Modul foreign, Version 0.5.2, doch
#  muessten die Beispiele auch unter aelteren
#  Versionen laufen)
##########################################################
#
# Es soll die SPSS-Datei nbg3.sav eingelesen
# werden; dabei handelt es sich um eine gekuerzte
# Fassung aus der Doktorarbeit von Andreas Klee
# ueber Lebensstile in der Stadt Nuernberg.
```


```
# Wer sich eingehender mit den Zielen und
# Ergebnissen der Arbeit vertraut machen will,
# sollte folgenden Artikel lesen:
#
#########
#
# ANMERKUNG P.R.:
#
# die Datei nbg.sav laesst sich von unserer
# Homepage aus beziehen
#         (siehe auch Links zu Referaten)
# unter
#
#    http://www.uni-bayreuth.de/departments/
#         math/org/mathe7/SPlus/referate/volkmar/
#         nbg.sav
#
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
########
#
#       Klee, Andreas (2001):
#       Der Raumbezug von Lebensstilen in der Stadt.
#       Ein Diskurs ueber eine schwierige Beziehung
#       mit empirischen Befunden aus der Stadt
#       Nuernberg
#       In: Muenchner Geographische Hefte, Band 83
#
# Andreas Klee hatte die Datei fuer seinen
# SPSS-Kurs als Uebungsdatei ins Intranet der Uni
# Bayreuth gestellt, sodass es statthaft er-
# scheint, sie HIER anzufuehren.
# Vor einer Weitergabe ausserhalb der Universitaet
# sollte man seine Genehmigung einholen.
#
# Zunaechst: Das Einlesen der Originaldatei
```


```
# nbg.sav mit R schlug fehl !!!
# (aus mir unbekannten Gruenden)
# Falls es fehlschlaegt, sollte man folgendes ver-
# suchen, was auch hier geholfen hat:
#
# * mit SPSS die Datei einlesen und als
#   "portable SPSS-Datei"
#   (Endung ".por") abspeichern.
# * Diese portable Datei kann dann (hoffentlich)
#   mit SPSS eingelesen werden.
# * Falls man die por-Datei unter SPSS erneut
#   einlaedt und als normale SPSS-Datei
#   (Endung ".sav" abspeichert, so laesst sich
#   diese seltsamerweise AUCH in R einlesen.
# (dies funktionierte mit SPSS Version 10; diese
#   ist auch ueber den Studentenserver USS der Uni
#   Bayreuth verfuegbar)
```


```
#
#
require(foreign)
# lade das R-Modul foreign, falls das nicht schon
# geschehen ist.
#
# Nun klappt das Einlesen mit:
nuernberg <- read.spss("nbg3.sav")
#
summary(nuernberg)
# Gibt einen Ueberblick ueber saemtliche Variablen
# und ihren Typ aus, und ob Level vorliegen.
#
# Leider werden die Variablen-Infos NICHT mit
# importiert, sondern nur die - in SPSS auf
# 8 Zeichen begrenzten und daher sehr
# kryptischen - Variablennamen. So kann man
```


```
# ohne weitere Information nicht erkennen,
# dass die Variable BESFREU kodiert, wie oft
# man Freunde besucht.
#
attach(nuernberg)
# Macht die Variablennamen sichtbar fuer R.
# ABER: ich rate davon ab! Falls naemlich ein
# Variablenname so heisst wie ein Befehl unter R,
# kommt es zu Namens-Konflikten
#
# Ohne attach() kann mit nuernberg[[3]] auf die
# (hier: 3.) Variable zugegriffen werden, dazu
# gleich mehr.
#
plot(ALLGZUST)
# Gibt ein Balkendiagramm aus, das die Anzahl der
# Faelle jedes Levels der Variable angibt, d.h.
```


```
# plot wirkt hier aehnlich wie ein Histogramm.
#
# Welche Level vorkommen, kann man abfragen mit:
levels(ALLGZUST)
# in diesem Fall gibt es 5 Level:
#     [1] "k.A."
#     [2] "Auffallend gepflegt"
#     [3] "Sauber und ordentlich,
#          normal"
#     [4] "Nicht besonders gepflegt, aber
#          halbwegs in Ordnung"
#     [5] "Heruntergekommen, etwas verwahrlost"
#
# Es gibt in R Probleme mit dem Unterstrich "_" in
# SPSS-Variablennamen, denn R interpretiert den
# Unterstich als "←" und liest anstatt der
# Variable C_ALTER demnach  C ← ALTER , was
```


```
# sinnlos ist.
# Dadurch kann man auf solche Variablen nicht
# direkt zugreifen.
# (M)ein folgender Ausweg ist umstaendlich:
# (kennt jemand einen eleganteren Weg?)
# (1):
try1 <- summary(nuernberg)
try2 <- as.list(try1[,1])
#
#
"C_ALTER" %in% names(try2)
# gibt TRUE oder FALSE aus, je nachdem der String
# "C_ALTER"  in der Liste der Variablennamen
# vorkommt oder nicht.
#
# (2):
try3 <- match("C_ALTER", names(try2))
```


```
# Ermittelt die Position, an der die Zeichenkette
# "ALTER" in der Liste der Variablennamen
# vorkommt, und weist es der Variable try3 zu.
# Liefert "NA", falls es nicht vorkommt.
#
# 3: Zuweisen von C_ALTER zur neuen Variable
# CALTER:  (es kodiert Altersklassen)
CALTER ← nuernberg[[try3]]
# wichtig sind dabei die doppelten eckigen
# Klammern. Bei einfachen Klammern, also
try4a ← nuernberg[try3]
# gehen dagegen die level verloren bzw. werden in
# Zeichenketten konvertiert.
#
# Beweis: folgender Befehl liefert "NULL". Wuerden
# die level erhalten bleiben, waere das Ergebnis
# "factor"
```


```
class(try4a)
#
# Anderer Beweis: Der folgende Befehl gibt die
# Anzahl der Level zurueck:
nlevels(try4a)
# das Ergebnis: 0 anstatt 6
#
# Nun koennen wir C_ALTER alias CALTER einmal
# plotten:
plot(CALTER)
# Leider beschriftet dies nicht alle Level.
#
plot(CALTER,SEX)
# Es wird ein interessantes zweifarbiges Balken-
# diagramm ausgegeben:
# * 1. Variable (hier: CALTER) wird als verschie-
#   dene Balken   interpretiert,
```


```
# * 2. Variable (hier: SEX) als farbige Abschnitte
#   auf diesen Balken. SEX kodiert uebrigens nur
#   das Geschlecht der Befragten ..
#
# vertauscht:
plot(SEX,CALTER)
# Geschlecht als Balken, Altersklassen als Farben.
# Kleiner Makel:
# Die Legende ueberschneidet sich mit den Balken
#
# andere Darstellungsweise: Sunflowerplot.
# Jeder Strich steht fuer eine "Date":
sunflowerplot(SEX,CALTER)
# Ist nur bei kleinen Datensaetzen gut, hier
# dagegen verschwimmen die "Sonnenblumen"
#
# Weitere Darstellungsweise:
```


```
try5 ← as.numeric(SEX)
try6 ← as.numeric(CALTER)
plot(jitter(try5),jitter(try6))
# Diese Darstellung ist ebenso wie Sunflowerplot
# gut geeignet, sehr selten vorkommende Level
# herauszufinden.
# Bei plot(CALTER,SEX) kann man solche Werte
# dagegen leicht uebersehen.
#
#
# ohne jitter, das ist die kleine zufaellige
# Verfaelschung, waere hier nichts aussagekraef-
# tiges zu sehen, denn alle Punkte desselben
# Levels werden uebereinander gezeichnet
# und erscheinen so als EIN einziger Punkt:
plot(try5,try6)
#
```


```
##################################################################
#
# In welche Datenstruktur hat R eigentlich die
# Datei eingespeichert?
# Dies findet man mit dem Befehl typeof() heraus:
typeof(nuernberg)
# hier: list
#
typeof(ALLGZUST)
# hier: integer
# finis
#
# Aehnlich arbeitet der Befehl mode()
mode(ALLGST)
# hier: numeric
```


R–Beispiel 1.3-10 [Import von ASCII-Dateien]:

```
# 2002 (c) Volkmar Klatt
# volkmar.klatt@stud.uni-bayreuth.de
#
##############################################
# Beispiel zum Einlesen einer Tabellen-Datei mit
# dem Statistikprogramm R
# (verwendet wurde hier die Version 1.4.1 von R,
#  doch muessten die Beispiele auch unter aelteren
#  Versionen laufen)
#
# Unter einer Tabellen-Datei wird hier eine Datei
# verstanden, die
#  * im reinen Text vorliegt
#    (ASCII = American Standard Code 2)
#  * Spalten (durch ein Trennzeichen getrennt) und
```


```
#  * Zeilen besitzt (getrennt durch
#      Wagenruecklauf-Zeichen)
#
# Statt langer Worte schaut man sich am besten
# 'mal die Datei ceuto.dat mit einem Texteditor
# an.
#
# Viele proprietaeren Dateiformate (die R nicht
# lesen kann), lassen sich in Tabellen-Dateien
# exportieren und so doch  noch einlesen.
#
# Warnung:
# Das deutsche Excel verwendet (voreingestellt)
# das Komma als  Dezimaltrennzeichen. Dann darf
# man beim Export in eine  Tabellen-Datei
# natuerlich NICHT das Komma als Trennzeichen
# waehlen, weil sonst Dezimalpunkt und Spalten
```


```
# ununterscheidbar werden. Empfehlenswert ist
# der Tabulator.
#########################################################
#
# Einlesen der Tabellen-Datei ceuto.dat
# Diese Datei kodiert das Frassverhalten von
#              Ceutorhynchus punctiger,
# (einem kleinen Kaefer), der im Loewenzahn lebt.
#
# An der Datenerhebung unter Leitung von
# Prof. emerit. Zwoelfer,  Universitaet Bayreuth,
# habe ich selbst mitgewirkt, sodass die
# (gekuerzten) Daten hier stehen koennen.
#
# Eine Weitergabe an Dritte sollte bitte
# nachgesucht werden!
#
```


```
#########
#
# ANMERKUNG P.R.:
#
# die Datei ceuto.dat laesst sich von unserer
# Homepage aus beziehen
#         (siehe auch Links zu Referaten)
# unter
#
#    http://www.uni-bayreuth.de/departments/
#         math/org/mathe7/SPlus/referate/volkmar/
#         ceuto.dat
#
##########
#
ceuto ← read.table("ceuto.dat", header=TRUE)
# header=TRUE bedeutet: die 1. Zeile der Datei
```


```
# soll so aufgefasst werden, als stuenden dort die
# Variablennamen.
# (was in ceuto.dat ja auch der Fall ist ;-)
# header=FALSE interpretiert auch die
# 1. Zeile als Daten
```


1.3.4 (g) Datenstruktur(en) der eingelesenen Daten in R

- **typeof**()
- modeof()
- siehe
  http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/referate/volkmar/spss_import.R

1.4 Arithmetik

1.4.1 implizites Casting

- so weit wie möglich natürliches Casting durch S: 2+T= 3, d.h. der Wert T (true) $\hat{=}1$, F (false) $\hat{=}0$
- Komplexe Arithmetik: nur bei Bedarf; muss z.T. erzwungen werden, vgl. **sqrt**(−2) und **sqrt**(−2+0i)

- Ist bei einer elementweise definierten mathematischen binären Operation einer der beiden skalar, so wird dieser Skalar elementweise als Operand verwendet, z.B. **seq**(0,5, **by**=1)∗3.
- für Arrays siehe auch Abschnitt 1.6.3

1.4.2 zyklisches Auffüllen

Sind bei einer binären Operation von zwei Vektoren diese nicht gleich lang, so wird der kürzere auf die Länge des längeren gebracht, indem der erstere zyklisch (möglicherweise unvollständig) wiederholt wird, z.B. **seq**(0,5, **by**=1)+**seq**(0,6,**by**=1).

1.4.3 einige Funktionen

- Grundrechenarten: +,−,∗,/, Bsp.:2+2 oder auch "+"(2,2)
- Potenzen, Wurzeln, Exp. und Log.: ^, **sqrt**, **exp**, **log**, **log10**
- (Hyperbel-)Trigonometrie und Inverse dazu: **sin**, **cos**, **tan**, **asin**, **acos**, **atan**, **sinh**, **cosh**, **tanh**, **asinh**, **acosh**, **atanh**

- Gamma-Funktion u.ä.: **gamma**(4):= $\Gamma(4)$, **lgamma**(4):= $\log_e \Gamma(4)$
- Rundung: **round**, **trunc**, ceil , **floor**
  round(x,3) rundet auf 3 Nachkommastellen, **round**(x,−3) rundet auf 10^3, **trunc**/ ceil / **floor** runden gegen 0/ab/auf.
- Teilen mit Rest: **%%**: modulo, **%/%** ganzzahlige Division
- Summen und Produkte: **sum**,**prod**,**cumsum**,**cumprod** bilden Summe und Produkt eines Vektors, resp. kumulative Versionen davon, i.e. z.B. **cumsum**(x)= $(\sum_{j\leq i} x_i)_{i=1,\ldots,n}$
- Minima und Maxima: **min**,**max**,**cummax**,**cummin**,**pmin**,**pmax**; erstere: (kumulative) Minima und Maxima eines Vektors; letztere: punktweise Minima und Maxima: x1= x_{1j}, x2,. . .,xm n-dimensionale Vektoren; dann ist **pmin**(x1 ,...., xm)= $(\min_i x_{ij})_j$
- Sortierung: **sort** ,**rev**; **sort** (x) sortiert den Vektor aufsteigend, **rev** gibt den Vektor in umgekehrter Reihenfolge

1.4.4 Auswertungsreihenfolge

[von höchster zu niedrigster Priorität]

$	auf Listenelement zugreifen
[, [[	auf Listen–/ Vektorelement zugreifen
^	exponenzieren
–	unäres Minus
:	Folgen erzeugen
%%,%\%,%∗%	und weitere spezielle Operatoren %...%
/,∗	multiplizieren, dividieren
+,–	addieren, subtrahieren
<,>,<=,>=,==,!=	vergleichen
!	logisch negieren
&, |,**&&**, ||	logische Operatoren
~	Formeln erzeugen
↞	innerhalb einer Funktion zuweisen, siehe Abschnitt 3.6
←,–>,_	zuweisen

1.4.5 Logische Ausdrücke

- können in offensichtlicher Weise aus den Operatoren <, >, <=, >=, ==, !=, !, &, | und Klammern gebildet werden
- **any** und **all** gut geeignet um logische Vektoren zusammenzufassen: geben einen einzigen logischen Rückgabewert zurück im Gegensatz zu den oben angeführten, die Vektoren von logischen Werten zurückgeben

1.4.6 Missings

- zwei Typen von Missings: NA (not available) und NaN (not a number) — (sowie Inf, −Inf)
- ternäre Logik: kein Vergleich x==NA möglich (liefert logischen Wert NA) $\rightsquigarrow$ **is.na**(x)

- Fehlerabfangen in eigenen Routinen durch **na.action**, **na. fail** , **na.omit**

1.5 String-Operationen

- anders als in C: character-Vektoren sind Vektoren von Strings!
- Unterscheide "" und **character**(0) $\rightsquigarrow$ unterschiedliches "leer"!
- Erzeugung durch Zuweisung $\leftarrow$"a"oder Verkettung **c**("a","a")
- lexikographische Ordnung: "a"<"bill"
- **nchar** liefert die Stringlänge der einzelnen Elemente, vgl. **nchar**(**c**("Hallo␣Rudie")) und **nchar**(**c**("Hallo␣", "Rudie"))
- **paste**
  - wandelt alle anderen Modi — Zahlen, Wahrheitswerte — in Strings
  - fasst beliebig viele Argumente zu einem Stringvektor zusammen

  - bei Bedarf auch zu einem String — Argument collapse
  - Bsp.:**paste**(**c**("Hallo", "␣Welt"), 1:3, 2==1+1)

- **grep**: im Index zur Mustererkennung — siehe Abschnitt 1.6.1 und Tabelle 1.10-1, z.B. alle Namen, die mit P anfangen und mit o enden namen[**grep**(pattern='^P.*o**$**',namen)]
- **substring**: Auswahl von Teil-Strings (siehe Beispiel)
- **abbreviate**: automatische Abkürzung der Strings (siehe Beispiel)
- **strsplit** : teilt $n \geq 1$ Strings in Unterstrings auf; Achtung Ergebnis ist Liste der Länge n, deren Elemente Vektoren mit aufgeteilten Strings sind
- **sub**, **gsub**: ersetzt reguläre Ausdrücke (vgl. Tabelle 1.10-1) — ersteres nur beim ersten Auftreten, letzteres überall

- R–Beispiel 1.5-1:

```
data(painters,package=MASS)
      # mittlerweile klar
pnames←row.names(painters)
      # Namen der Maler in pnames
substring(pnames[1:4],1,5)
as.vector(abbreviate(pnames[1:4]))
```

1.6 Indizes und Arrays

1.6.1 Indizierungsmöglichkeiten

(i) mit logischen Vektoren:

nur die Elemente werden verwendet die eine logische Bedingung erfüllen

Bsp: `z←(x+y)[!is.na(x) &x>0]`

(ii) Vektoren mit positiven Integern

Diese Integers müssen im Bereich '0:length(x)' liegen

Bsp: `x[c (1:3,5,10:13,5:1)]`

(iii) Vektoren mit negativen Integern:

Die entsprechenden Elemente werden herausgenommen

Bsp: `x[-(1:3)]`

(iv) Vektor mit Charakter-Strings

Auswahl anhand von Namen, z.B.

```
obst<-c(5,10,1)
names(fruit)<-c("Orange","Birne","Apfel")
essen<-obst[c("Apfel","Orange")]
essen
```

BEMERKUNG 1.6-1:

Zuweisungen: Auch Zuweisungen können so vorgenommen werden,

`x[is.na(x)]<-0 # ersetzt alle Missings durch 0`

unzul. Indizes:

- ergeben Fehler bei *Ausdrücken*
- bei Zuweisungen $\rightsquigarrow$ Missings, falls ≥ 0, sonst ignoriert

`replace, append`:
erzeugen jeweils Kopien des Arguments, ohne das Original zu verändern;

- **replace**(x, **pos**, values) erzeugt Kopie mit x[**pos**]$\leftarrow$values,
- **append**(x,values, after) erzeugt Kopie mit angehängten Werten values ab Position after

1.6.2 Arrays und Indizierung

- *Array*: multi-indiziertes Variablenschema;
  bei 2 Indizes $\rightsquigarrow$ *Matrix*
- Dimensionen der einzelnen Indizes: im Vektor **dim** abgelegt
- Nummerierung startet mit 1 — nicht wie in C mit 0
- Indizierung: wie 1.6.1 (i), oder mit mehrdimensionalen Indizes wie in 1.6.1 (i)–(iv)
- Erzeugen einer Matrix z.B. durch **matrix**(0,**nrow**=3,**ncol**=4)
- bei Casting *Vektor* $\leftrightarrow$ *Array* "umgekehrtes Stellenwertsystem", d.h. [1,2] entspricht größerer eindim. Index als [2,1]
  — Formel für **dim**$=(d_1,\ldots,d_m)$

$$[i_1,\ldots,i_m] \leftrightarrow [i_1 + \sum_{j=2}^{m} \left\{ (i_j - 1) \prod_{k=1}^{j-1} d_k \right\}]$$

- Namen für die Indizes: **dimnames**, z.B. a←**array**(a,**dim**=**c**(3,4,10)), **dimnames**(a)←**list**(letters [1:3], **c**("i","ii","iii","iv"),NULL)
- Permutieren der Indizes durch **aperm**, z.B. **aperm**(a,(2,3,1))
- weitere Indizierungsmöglichkeiten

  (i) jede beliebige Indexposition darf leer bleiben:
  - entsprechender Index wird ganz durchlaufen
  - Zahl der Indizes wird um einen kleiner
  - keine Veringerung der Indexzahl $\rightsquigarrow$ sa←a [2,,, **drop**=F]
  - **drop** gibt es nicht bei Matrizen

  (ii) *Array* indiziert durch *Matrix*

  bei m Indexdimensionen können jeweils k Elemente durch eine $k \times m$ Matrix ausgewählt werden.

1.6.3 Arithmetik mit *Arrays*

— *Arrays* als Input für Funktionen mit skalaren Argumenten —

- Einsetzen von einheitl. dim. *Arrays*:
  liefert elementweise ausgewertete Funnktion zurück, z.B.
  `sin(matrix((1:10)*2*pi/10,nrow=5,ncol=2))`
- Einsetzen von einheitl. dim. *Arrays* und Vektoren:
  - Auswertung von links nach rechts
  - Erweiterung zu kurzer Vektoren gem. Abschnitt 1.4.2
  - Arrays müssen von einheitlicher Dimension sein
  - Ist ein Vektor länger als die vorangegangenen Arrays so ist das Resultat ein Vektor dieser längeren Dimension
  - Liegen Arrays vor, und gab es weder Casting in einen Vektor noch einen Fehler, so ist das Resultat ein Array mit den gemeinsamen Dimensionen

1.6.4 Sortieren

- kanonisch mit **sort**
- dabei: Reihenfolge von mit "==" verglichenen Elementen bleibt erhalten
- partielles Sortieren möglich
- flexibler: i←**sort** . **list** (x): liefert Indexvektor 'i' zurück, so dass 'x[i]' sortiert ist
- ebenfalls die Funktion **order** mit ihr simultanes Sortieren nach mehreren Kriterien (z.B. erst Name dann Vorname)
- Ränge: die Funktion **rank**

1.7 Matrix–Operationen

BEMERKUNG 1.7-1:
Das meiste in Abschnitt 1.7 gilt auch für numerische *Data-Frames*.

Im folgenden seien `X`, `X1`, `X2` Matrizen und `y`, `y1`, `y2` Vektoren, so dass die Dimensionen immer "passen".

1.7.1 Anhängen von Spalten und Zeilen

- Anhängen von Zeilen durch **rbind**, von Spalten durch **cbind**
- Ist der anzuhängende Vektor zu kurz, wird er wie in Abschnitt 1.4.2 aufgefüllt.

1.7.2 Matrixprodukte und Transposition

- Transposition: **t**(X),**t**(y)
- Matrixmultiplikation: X1%∗**%**X2, X1%∗**%**y, y%∗**%**X1
  Beachte: Bei Linksmultipl. mit Vektor keine Transposition!
  Genauer: %∗**%** inneres Produkt $\Rightarrow$ x%∗**%**x$= \|x\|_2^2$
- Kreuzprodukt: **crossprod**(X1,X2)$= X_1^\tau X_2$
- äußeres Produkt: y1%o%y2$= y_1 y_2^\tau$ (dyadisches Produkt),
  X1%o%X2$= X_1 \otimes X_2$ (Kroneckerprodukt)
- Funktion **outer**: für $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ und $f : \mathbb{R}^2 \to \mathbb{R}$ gibt
  outer(y1,y2,f) $(f(x_i, y_j))_{i=1,\ldots,m,j=1,\ldots,m}$.

R–Beispiel 1.7-2 [Faltung von Bin(4,0.3) und Bin(3,0.1)]:

```
# X~Bin(4,0.3) und Y~Bin(3,0.1) unabh.
# gesucht: die Wahrsch.fkt. von X+Y
x<-dbinom(0:4,size=4,p=0.3)
    # dbinom(..)=P(X=x)
y<-dbinom(0:3,size=3,p=0.1)
    # dbinom(..)=P(Y=y)
xy<-x%o%y    # ergibt pi*qj
ind<-outer(0:4,0:3,"+")
pf<-function(i){
      ind0<-(ind==i)
      return(sum(ind0*xy))}
c(pf(0),pf(1),pf(2),pf(3),pf(4),pf(5),pf(6),pf(7))
```


1.7.3 apply und sweep

1.7.3 (a) apply

- **apply** erlaubt es, skalare-argumentige Funktionen sukzessive, schnittweise auf *Arrays* auszuwerten
- vermeidet Schleifen; falls Matrixmultiplikation möglich, ist diese aber schneller
- Argumente
  - Name des *Arrays*
  - Integer-Vektor `MARGIN`, der die Indizes festlegt, auf die die Funktion separat angewendet werden soll
    Beachte: das Resultat hat Dimensionen **dim**(X)`[MARGIN]`
  - Name einer Fkt., `FUN`, die auf die Schnitte angewendet wird
  - alle weiteren Argumente von `FUN`
  - entsprechende Varianten für andere Strukturen: **sapply**, **lapply**, **tapply**, c.f. Abschnitte 1.8.5 und 1.8.3

- R–Beispiel 1.7-3:

```
data(iris3) # laden des IRIS- dim(iris3)
            # Datensatzes
help(iris3) # Info dazu
ir.means<-apply(iris3, c(2,3), mean)
            # Anwendung von mean auf jeden
            # j,k schnitt von iris[i,j,k]
apply(iris3, c(2,3), mean, trim=0.1)
      # zus. Arg von mean -> getrimmtes Mittel
apply(iris3, c(2), mean)
ir.var<-apply(iris3,3,var)
      # Varianz eines jeden k-Schnitts
```

1.7.3 (b) sweep

Bereinigen der Daten — im obigen Beispiel Abzug des Mittelwerts

```
sweep(iris3,c(2,3), ir.means)
```


1.7.4 Funktionen in Matrizen

- Lösen linearer Gleichungssysteme: **solve**
  solve(A) invertiert (A),
  solve(A,b) berechnet ein x, so dass $Ax = b$,
  bei überbestimmtem A die KQ-Lösung
- Choleskizerlegung: **chol**, **backsolve**
- Eigenwerte / –vektoren: **eigen**
  liefert Liste mit Komponenten values (Eigenwerte) und vectors (Eigenvektoren)
  nur –werte mit only.values=T,
  deklarativ symmetrisch symmetric=T
- Singulärwertzerlegung **svd**
- QR-Zerlegung **qr**
- Determinante: nicht direkt;
- Spur: nicht direkt; $\rightsquigarrow$ mit **qr** oder **eigen** Achtung **nicht zu verwechseln mit** **trace**, siehe Abschnitt 3.4.3 (c)

1.7.5 Casting für *Matrizen* und *Data-Frames*

- *Matrix* → *Data-Frame* **as.data.frame**; Namen übernommen oder Defaults
- *Data-Frame* → *Matrix*

  as.matrix: sind Strings im *Data-Frame* ⇒ Modus: character (alle, auch numerische Spalten!)

  data.matrix : Modus: numeric (Strings werden auf numeric gecastet!), ursprgl. Strings in column.**levels**

1.8 Funktionen von Faktoren und Listen

1.8.1 ein hypothetisches Datenbeispiel

R–Beispiel 1.8-1:

```
land <- c("HHA","BAY","NRW","BAW","RLP","HBR",
          "BAY","SAC","BAW","SAA","HES","NS")
```


```
selbst <- c(T,T,F,T,F,F,F,T,F,T,T,T)
EKT <- c(30,28,23,40,14,40,24,90,29,30,60,40)

Beruf.studie <- data.frame(land, selbst, EKT)
rm(land, selbst, EKT)
attach(Beruf.studie) # siehe Abschnitt 1.10.3 (c)
```

1.8.2 table und tabulate

- im Beispiel werden die Strings und logischen Werte zu *Faktoren*
- **table** listet die absoluten Häufigkeiten eines jeden Faktors, in Beispiel 8.1-1: **table**(EKT), **table**(selbst)
- bei mehr als einem Argument bildet **table** Mehrfach-Kreuztabellen, in Beispiel 8.1-1: **table**(EKT,selbst)
- die Funktion **tabulate** listet die absoluten Häufigkeiten in einem Vektor mit nicht negativen, numerischen Einträgen

- es entsteht ein *ragged array*, eine Liste mit heterogenen unterschiedlich langen Einträgen
- ist das erste Argument der Funktion ein *Faktor*, so erzeugt die entsprechende Methode **plot**, i.e. **plot . factor**, *boxplots*, siehe auch Abschnitt 4.2

1.8.3 tapply

- wie in Abschnitt 1.7.3 können `for`–Schleifen vermieden werden, hier durch **tapply**,
- in Beispiel 1.8-1: **tapply**(EKT,land,**mean**)
- Argument 1: zu betrachtende Variable, Argument 2: (Liste von) klassierende(n) Variablen, Argument 3: anzuwendende Funktion / Statistik

1.8.4 split

- **split** splittet den Datensatz gemäß einer kategoriellen Variable auf
- in Beispiel 1.8-1: **split** (EKT,selbst)

1.8.5 lapply und sapply

- die Analoga zu **apply** und **tapply** für Listen sind **lapply** und **sapply**
- sind alle Einträge gleich lang, so vereinfacht **sapply** das Resultat zu einer Matrix / einem Vektor
- in Beispiel 1.8-1: **sapply**(**split** (EKT,selbst),**mean**)

1.9 Datenausgabe

1.9.1 Ausgabe auf File

- der Befehl **write**
  - Syntax: **write**(<objname>,file="<filename>")
  - schreibt Objekt <objname> auf das File <filename>; man beachte auch Bemerkung 1.3-2
  - <filename> entspricht schreiben auf das Sitzungs–Fenster
  - Argument ncolumns spezifiziert die Spaltenzahl pro Zeile
- der Befehl **write.table**, z.B.
  write.table(painters , **file** ="",sep="\t::␣")
  - per default werden Zeilen– und Spalten–Labels mit geschrieben — so vorhanden
  - **row.names**=F,**col.names**=F lässt beide weg

- der Befehl **write.matrix** von Venables and Ripley (1999)

```
write.matrix<-function(x,file="",sep=" ")
{
  x<-as.matrix(x)
  p<-ncol(x)
  cat(dimnames(x)[[2]],format(t(x)),file=file,
      sep=c(rep(sep,p-1),"\n"))
}
```

- der Befehl **cat**
  - ähnlich wie **paste** mit Argument `collapse=""`
  - explizites `\n` zum Zeilenumbruch nötig
  - Argument `fill=<number>` $\rightsquigarrow$ `<number>` Zeichen pro Zeile; `fill=T` bricht genau auf Fensterbreite um
  - Argument `labels=letters` kennzeichnet die Zeilen durch Buchstaben
  - Argument/Funktion **format**

    - zwingt die Ausgabe auf bestimmtes Format, vgl. Format-String in C — `printf`, `scanf`
    - Beispiel: Deklaration der Funktion **print.summary.lm**

- der Befehl **dump**
  - Syntax: **dump**(`"<objname>",file="<filename>"`)
    bzw.
    dump(**list**(`"<obj1>","<obj2>",...`),`file="<fname>"`)
  - schreibt [die Liste von] Objekt[en] `<obj[name][..]>` als Zuweisungen auf das File `<f[ile]name>` $\rightsquigarrow$ gut lesbar;
  - mit **source**(`"<filename>"`) wieder einlesbar (aber langsam)
  - schneller mit **dump.data** und **data.** restore
    - zum Übermitteln von Datensätzen zwischen (verschiedenen) Rechnern;
    - Speichermodus wird mit übermittelt

- der Befehl **save**
  - Syntax: **save**("<objname>",file="<filename>")
    bzw.
    dump(**list**("<obj1>","<obj2>",...),file="<fname>")
  - weitere Optionen:
    * ascii : falls auf TRUE wird in ASCII–Code abgelegt; per default auf FALSE; dann abspeichern in einem plattformunabhängigen Binärformat (XDR–Darstellung)
    * compress: falls auf TRUE werden die Daten komprimiert abgelegt; per default auf FALSE;
  - schreibt [die Liste von] Objekt[en] <obj[name][..]> als ASCII– oder Binärdatei auf das File <f[ile]name> $\rightsquigarrow$ komprimierten Files
  - Daten schnell wieder in R mit **load**("<filename>") unabhängig vom Betriebssystem einlesbar;
  - Variante: **save.image**(file="<filename>")

- schreibt den gesamten Arbeitsspeicher als ASCII– oder Binärdatei auf das File <f[ile]name>; siehe auch Abschnitt 8.2.1 (a)
- zusätzlicher Parameter:
  safe: falls auf TRUE wird zuerst auf eine temporäre Datei geschrieben, und diese erst nach erfolgreichem Abschluss des Speicherns in "<filename>" umbenannt

1.9.2 Umleiten der Ausgabe / Drucken

- Mit dem Befehl **sink**("<filename>") wird die Bildschirmausgabe auf das auf das File <filename> umgelenkt
- zum Drucken: am besten Umweg über Ausgabefile, das nacheditiert werden kann
- zum Ausdruck von Graphik siehe Abschnitt 4.1.2

1.10 Arbeiten mit dem System

1.10.1 File- und URL-zugriffe unter R

- um systemunabhängig auf Files zugreifen zu können: siehe ? files
- **file** . **create** (...) , **file** . **exists** (...) , **file** . **remove** (...) erzeugt File/testet es auf Existenz/löscht es
- dabei: Argumente ... durch "/" getrennt zu Pfad zusammengehängt
- mit **file** . rename, **file** . copy, **file** . symlink(from, to) kann man Files umbenennen, kopieren, bzw. Verknüpfungen erstellen
- **file** . **append** hängt Files zusammen
- **dir** . **create** erzeugt einen Ordner
- **dir** (synonym: **list** . files) gibt den Inhalt eines Ordners aus

genauer: das systemspezifische Ordner-Trennzeichen

- **download.file** lädt eine URL in ein entsprechendes temporäres File; dabei sind für Proxies eventuell Systemvariablen mit Sys.putenv zu setzen —vgl. auch ?**download.file**, ?Sys.putenv
- das aktuelle Arbeitsverzeichnis erhält man mit **getwd** und setzt es mit **setwd**

1.10.2 R-Skripte

- R-Skripte sind ASCII–Files, vorzugsweise mit Endung `.r`, mit Folgen von R-Anweisungen
- werden dann Zeile für Zeile abgearbeitet (interpretiert !)
- Einlesen eines Sripts mit **source**`("<filename>")`,
  — möglicherweise mit **options**`(echo=T)`
- Bearbeitung mit einem Editor Ihrer Wahl (siehe auch Übung)
- zum Erstellen oft nützlich: das Editieren von History-Files, vgl. Abschnitt 1.10.5

1.10.3 Auffinden von S-Objekten

1.10.3 (a) Daten–Filestruktur in S

- interaktiv erzeugte Objekte in S-Plus real, in R virtuell als File abgelegt:

S-Plus unter (.Data bzw. _DATA)–Verzeichnis

R im virtuellen Verzeichnis ".GlobalEnv"

- Objekte, die auf einer höheren Ebene generiert werden, — z.B. lokale Variablen in einem Funktionenkörper, — werden in einem lokalen *frame* abgelegt (vgl. auch Abschnitt 3.6)
- jedes Objekt wird in einem separaten File abgelegt (auch mit cp/copy, rm/del etc. manipulierbar)
- in S-Plus: alle Objekte aus früheren Sitzungen bleiben verfügbar (als Files)
- besser $\rightsquigarrow$ mehrere Arbeitsverzeichnisse für unterschiedliche Projekte

1.10.3 (b) Suchmechanismus in S

- auf der Suche nach einem Objekt durchsucht S zunnächst die *Search list* (Suchliste)
- diese Liste erhält man mit **search**()
- die Elemente des ersten Elementes der Suchliste erhält man mit **objects**(), die der weiteren mit **objects**(<number>)
- durch weiteres Argument pattern (mit Wildcards) Einschränkung der Suche
- Wildcard-Syntax gemäß Unix–Standard POSIX 1003.2, siehe auch Tabelle 1.10-1
- zu einem Objekt <obj> findet **find**(<obj>) die Stellen in der Suchliste

Tabelle 1.10-1 [Reguläre Ausdrücke]:

— aus Aho et al. (1988),fig. 3.48

Ausdruck	wird gematch-t durch	Beispiel
c	jedes non-operator Zeichen c	a
\c	Zeichen c als solches	*
"s"	String s als solches	"**"
.	jedes Zeichen außer newline	a.*b
^	Zeilenanfang	^abc
$	Zeilenende	abc$
[s]	jedes Zeichen in s	[abc]
[^s]	jedes Zeichen, das nicht in s ist	[^abc]
r*	0 oder mehr r's	a*
r+	1 oder mehr r's	a+
r?	0 oder mehr r	a?
r{m,n}	m bis n–maliges Auftreten von r	a{1,5}
r1r2	r1 gefolgt von r2	ab
r1|r2	r1 oder r2	a|b
(r)	r	(a|b)
r1/r2	r1 falls es von r2 gefolgt wird	abc/123

- Einträge der Suchliste heißen *Dictionary* oder *Database*
- neben S–Files können auch Listen oder listen–artige Objekte wie Data-Frames Einträge sein
- an erster Stelle

S-Plus `.Data` bzw. `_DATA`)–Verzeichnis;

R `.GlobalEnv`

dieses heißt auch Arbeitsverzeichnis (Workspace)

- mehrere Objekte mit gleichen Namen: das im "früheren" Directrory (in Suchliste) maskiert die "dahinter" kommenden
- spezifische Variante aus Directory `<lstnummer>` mit **get**`("<objname>",<lstnummer>)`
- Hinzufügen weiterer directories, Listen oder Data-Frames (in die Suchliste) mit **attach**
  hierbei Erstellen einer Kopie (keine Auswirkung auf "Original")

- Löschen weiterer directories, Listen oder Data-Frames (aus der Suchliste) mit **detach**
  Modifikationen in lokaler Kopie werden in File von der Gestalt `Save.<objname>.<lstnummer>` — außer Argument `save=F` in **detach**
- Beträchtliches Caching $\rightsquigarrow$ logische Position (in Suchliste) $\not\in$ Position im Filesystem; Abgleich mit synchronize

1.10.4 Systemkonfiguration

- vielfältige Anpassungsmöglichkeiten des Systems $\rightsquigarrow$ sogar Modifikation von Systemfunktionen
- der Befehl **options**,
  - z.B. **unlist (options)**
  - Auskunft über Option `<optname>` mit **options**`("<optname>")`, z.B. **options**`("prompt")`

  - Setzen der Option <optname> auf Wert <optwert> mit **options**(<optname>=<optwert>), z.B. **options**(prompt=":")
  - wichtige Optionen siehe Tabelle 1.10-2

- ähnlich: **ps.options**, siehe Abschnitt 4.1.2 (b)
- der Befehl .First
  - wird beim Aufruf von R "still" ausgeführt (falls in .Data bzw. _DATA vorhanden)
  - Beispiel aus Venables and Ripley (1999)

```
.First ← function()
{
options(prompt="> ", continue="+ ",
        digits=5, length= 99999)
ps.options(paper="a4", font=3,
        pointsize=10, horizontal=F)
library(MASS, first=T)
```


```
}
```

- der Befehl .Last
  - analog .First wird bei Abschluss der Sitzung von R “still” ausgeführt
  - Beispiel (c.f. Venables and Ripley (1999))

```
.Last←function(){cat("und␣tschuess ....\ n")}
```


Tabelle 1.10-2 [wichtige Optionen]:

`width`	Seitenweite, in `char`–Symbolen
`length`	Seitenlänge, in `char`–Symbolen, oft zum Auftrennen langer Ausgaben
`digits`	signifikante Zahl an Stellen bei Ausgabe mit **print**
`editor`	der von Ihnen präferierte Editor
`echo`	logischer Wert: Sollen Ausdrücke vor Auswertung noch ausgegeben werden?
`prompt`	primärer Prompt (per default >)
`continue`	Kommando–Fortsetzungs–Prompt (per default +)
`error`	Funktion zum Fehlerabfangen (siehe Abschnitt 3.4)
`warn`	Strenge bei der Behandlung von Warnungen; `0` sammelt sie und gibt sie gemeinsam aus, `1` gibt sie sofort aus, `2` macht Warnungen zu Fehlern
`memory`	maxim. Speicher (in bytes) der alloziert werden kann
`object.size`	maximaler Speicher (in bytes) eines Objekts

1.10.5 History–File

- mit history(max.show = 25, reverse = FALSE) werden die ersten [letzten] max.show in dieser Sitzung ausgeführten Befehle in chronologischer [umgekehrter falls reverse = T] Reihenfolge gelistet
- Abspeichern der Historie mit savehistory(<filename>), per default in .Rhistory
- Laden einer Historie mit loadhistory(<filename>)

2 Einfache explorative Analyse

2.1 etwas Stochastik/Statistik

Dieser Abschnitt stellt im Range einer Wiederholung wichtige Begriffe und Verteilungen aus der Stochastik I zusammen, führt die Notation ein, und geht dann auf deren Umsetzung in R ein

2.1.1 Wahrscheinlichkeitsmaße

- mathematische Umsetzung der Information / Entscheidungsgrundlage: die *σ–Algebra* $\mathcal{A}$
  - Teilmenge der Potenzmenge einer Ereignismenge Ω, $\Omega \in \mathcal{A}$
  - abgeschlossen unter $\cdot^c$ und abzählbar vielen $\cap$, $\cup$
- axiomatischer Wahrscheinlichkeitsbegriff: ein *Wahrscheinlichkeitsmaß* gemäß Kolmogoroff'schen Axiomen
  - $P : \mathcal{A} \to [0, 1]$, $P(\Omega) = 1$

  - σ–additiv: $\{A_i\}_{i\in\mathbb{N}} \subset \mathcal{A}$ disjunkt $\Rightarrow P(\bigcup_i A_i) = \sum_i P(A_i)$
  - $(P(A \cap B) + P(A \cup B) = P(A) + P(B)$ für alle $A, B \in \mathcal{A})$

- Paar $(\Omega, \mathcal{A})$ heißt *Messraum*
- Tripel $(\Omega, \mathcal{A}, P)$ heißt *Wahrscheinlichkeitsraum* (W-Raum)

2.1.2 Zufallsvariablen und Verteilungen

2.1.2 (a) Zufallsvariablen

- eine Abbildung $X : \Omega_1 \to \Omega_2$ zwischen zwischen Messräumen $(\Omega_1, \mathcal{A}_1)$, $(\Omega_2, \mathcal{A}_2)$ ist *messbar*, falls $X^{-1}(A_2) \in \mathcal{A}_1$ für alle $A_2 \in \mathcal{A}_2$;

  anschaulich: Ich kann mit der Information aus $\mathcal{A}_1$ entscheiden, welche Werte X in $\mathcal{A}_2$ annimmt.

- jede messbare Abbildung von einem W–raum $(\Omega_1, \mathcal{A}_1, P)$ in einen Messraum $(\Omega_2, \mathcal{A}_2)$ heißt *Zufallsvariable* (ZV).
- der Messraum $(\Omega_2, \mathcal{A}_2)$ heißt auch Stichprobenraum

- i.a. genaue Gestalt von $(\Omega_1, \mathcal{A}_1, P)$ unwichtig, es interessiert nur das Realisationsverhalten von X in Ω_2.
- Beispiele für $(\Omega_2, \mathcal{A}_2)$:
  - Realisationen eines Würfels
  - Antworten in einem Fragebogen
  - Kurvenverlauf des DAX

2.1.2 (b) Mess–Skalen

- je nach Stichprobenraum nur bestimmte Operationen mit den Daten zulässig;
- Klassifikation durch *Skalenniveau*
  - kategorielle Merkmale: z.B. Geschlecht, Marktsegment, Augenfarbe, Landschaftstyp
    nur relative Häufigkeiten, z.B. Modus

  - ordinale Merkmale: z.B. Schulnoten, "stimme-zu-Skalen"
    Anordnung zulässig $\Rightarrow$ auch Ordnungsstatistiken wie Median, Quartile zulässig
  - metrische Merkmale: z.B. Alter, Einkommen,
    auch: Unterscheidung Ratio–/ und Intervallskalen
    Addition (und bei Ratio-Skalen auch Division) zulässig $\Rightarrow$ auch Mittelwert, Varianz zulässig

- abgeleitete Skalen: Vektor–/ "Listen"–wertige Variablen

2.1.2 (c) Verteilungen/Dichten

- Information über Realisation einer einzelnen ZV vollständig beschrieben durch Bildmaß $P^X : \mathcal{A}_2 \to [0,1]$,
  $P^X(A_2) = P(\{\omega_1 \in \Omega_1 \mid X(\omega_1) \in A_2\})$ für $A_2 \in \mathcal{A}_2$,
  statt P^X auch $\mathcal{L}(X)$
- $\mathcal{A}_2$ ziemlich groß, stattdessen repräsentative Unterklassen von Ereignissen

- für diskrete Merkmale: alle Elementarereignisse $\{\omega_{2;i}\}$ in Ω_2, $\rightsquigarrow$ W-Funktion $\tilde{P} : \Omega_2 \to [0,1]$, zum Beispiel durch Angabe der W-keit für Ereignisse vom Typ {"6"} beim Würfeln
- für metrische Merkmale mit Werten in $\mathbb{R}$ oder $\mathbb{R}^k$: Ereignisse $(\infty; x]$ in $\mathbb{B}$ $[\mathbb{B}^k]$, $\rightsquigarrow$ Verteilungsfunktion $F^X : \mathbb{R}^k \to [0,1]$, $F(x) = P(X \leq x)$; Konvention: im $\mathbb{R}^k$ heißt $x \leq y$ falls $x_i \leq y_i$ für alle i.

- bei (absolut)stetigen Merkmalen:
  Falls $\lambda \gg P^X$ besitzt $\mathcal{L}(X)$ eine (Lebesgue–)Dichte, also $\exists p \in L_1(\lambda), p \geq 0\ [\lambda]$, so dass für alle $A \in \mathcal{A}_2$

$$P(X \in A) = \int_A p \, d\lambda = \int_A p(x) \, dx$$

  dabei heißt $p \geq 0\ [\lambda]$: $\lambda(p < 0) = 0$

2.1.3 wichtige Verteilungen

2.1.3 (a) diskrete Verteilungen

- *Einpunktverteilung / Diracmaß* in x, in Zeichen $\mathrm{I}_{\{x\}}$
- *Uniforme Verteilung* auf einer endlichen Menge von Elementen a_i, in Zeichen $\mathrm{ufo}(\{a_1, \ldots, a_n\})$
- *Bernoulli–Verteilung*: $P(X = 1) = p = 1 - P(X = 0)$; Parameter p, in Zeichen $\mathrm{Bern}(p)$
- *Binomial–Verteilung*: Vtlg. der Summe von n u.i.v. Bernoulli–Variablen; n-fache Ziehung mit Zurücklegen; Parameter n, p, in Zeichen $\mathrm{Bin}(n, p)$
- *Hypergeometrische Verteilung:* n-fache Ziehung ohne Zurücklegen aus Urne mit N Kugeln, k weiß, $N - k$ schwarz; Parameter N, n, k, in Zeichen $\mathrm{HypGeo}(N, n, k)$

- *Multinomial–Verteilung:* Vtlg. der Summe von n u.i.v. Variablen mit m Merkmalsausprägungen; Parameter n, $p_1, \ldots, p_{m-1}$, in Zeichen $\mathrm{Multnom}(n, p_1, \ldots, p_{m-1})$
- *Wilcoxon–Verteilung:* Vtlg. der entsprechenden Teststatistik, Parameter n, m, in Zeichen $\mathrm{Wilcox}(n, m)$
- *Poissonverteilung:* Grenzwert für "seltene Ereignisse", Parameter λ, in Zeichen $\mathrm{Poiss}(\lambda)$
- *Geometrische Verteilung:* $P(X = x) = p(1-p)^{x-1}$, $x \in \mathbb{N}_0$, Parameter p, in Zeichen $\mathrm{Geo}(p)$
- *negative Binomial–Verteilung* (auch *Pascal–Verteilung*): Vtlg. der Wartezeit, bis man bei u.i.v. $\mathrm{Bern}(p)$–Variablen zum n-ten Mal "1" erzielt; Parameter n, p, in Zeichen $\mathrm{negBin}(n, p)$; Spezialfall $(n = 1)$: Geom. Vtlg;

2.1.3 (b) die Normalverteilung

- wichtig wegen Zentralem Grenzwertsatz
- Parameter: μ, σ^2, in Zeichen $\mathcal{N}(\mu, \sigma^2)$
- multivariat: Seien $X_1, \ldots, X_m \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0,1)$;
  fassen wir diese zu X zusammen, und ist $A \in \mathbb{R}^{p \times m}$, $\mu \in \mathbb{R}^p$, so heißt die Verteilung von $Y = AX + \mu$ *multivariat normal* mit Parametern μ, $\Sigma = AA^\tau$, in Zeichen $\mathcal{N}_p(\mu, \Sigma)$

2.1.3 (c) von der Normalverteilung abgeleitete Verteilungen

Im folgenden sei $X = (X_1, \ldots, X_m)^\tau \sim \mathcal{N}_m(\mu, \mathbb{I}_m)$,
$\mu = (\mu_1, \ldots, \mu_m)^\tau$, $\|\mu\|^2 = c$, $Y = X - \mu$;

- $\mathcal{L}(\exp(Y_1))$: *Lognormal–Vtlg*
- $\mathcal{L}(\|X\|^2)$: $\chi^2_m(c)$ oder *Chiquadrat–Vtlg* mit m Freiheitsgraden (df's) und Nichtzentralität (ncp) c

- $\mathcal{L}(Y_1/Y_2)$: Cauchy–Vtlg — ohne Erwartungswert!
- Sei $\tilde{X} \sim \mathcal{N}(\tilde{\mu}, 1)$ von X sto.u.;
  dann $\mathcal{L}(\sqrt{m}\tilde{X}/\|Y\|)$: $t_m(c)$ / *Student–Vtlg* mit m df's und ncp c.
- Sei $\tilde{X} \sim \chi^2_p$ von X sto.u.;
  dann $\mathcal{L}(\frac{\|X\|^2/m}{\tilde{X}/p})$: $F_{m,p}(c)$ / *Fisher–Snedecor–Vtlg* mit df's m und p und ncp c.
- Seien $Z_1, \ldots, Z_n \overset{\text{u.i.v.}}{\sim} \mathcal{N}_m(0, \Sigma)$;
  fassen wir die Z_i zu einer Matrix $Z \in \mathbb{R}^{m \times n}$ zusammen,
  dann $\mathcal{L}(ZZ^\tau)$: $\text{Wish}_m(n, \Sigma)$ / *Wishart-Vtlg* mit Parametern n, Σ
- Sei $U \sim \mathcal{N}_m(0, \mathbb{I}_m)$ sto. u. von $Z \sim \text{Wish}_m(n, \mathbb{I}_m)$;
  dann $\mathcal{L}(nU^\tau Z^{-1} U)$: $T^2_m(n)$ oder *Hotelling–Vtlg*.
- Seien $\tilde{Z} \sim \text{Wish}_m(p, \Sigma)$ und $Z \sim \text{Wish}_m(n, \Sigma)$ sto. u.
  dann $\mathcal{L}(\frac{\det \tilde{Z}}{\det(\tilde{Z}+Z)})$: $\Lambda_m(p, n)$ oder *Wilks–Vtlg*.

2.1.3 (d) sonstige stetige Verteilungen

- *Uniforme Verteilung* auf $[a, b]$ in Zeichen $\mathrm{ufo}([a, b])$
- *Exponential–Verteilung:* gedächtnislose Verteilung, Parameter λ, in Zeichen $\mathrm{Exp}(\lambda)$
- *Laplace–Verteilung:* symmetrisierte Exponentialverteilung mit Parameter λ, in Zeichen $\mathrm{Lapl}(\lambda)$.
- *Gamma–* oder *Erlang–Verteilung:* Verteilung mit Parametern $\alpha, \lambda > 0$ und Vtlgsfkt. $t \mapsto \lambda^\alpha \Gamma(\alpha)^{-1} \int_0^t x^{\alpha-1} e^{-\lambda x}\, dx$, in Zeichen $\mathrm{Gam}(\alpha, \lambda)$;
  falls $\alpha \in \mathbb{N}$: Vtlg. der Summe von n u.i.v. $\mathrm{Exp}(\lambda)$–Variablen.
- *logistische Verteilung:* $P(X \leq x) = (1 - e^{-x})^{-1}$, in Zeichen logist.
- *Beta–Verteilung:* Vtlg. der k-ten Ordnungsstatistik von n u.i.v. $\mathrm{ufo}([0, 1])$–Variablen, in Zeichen $\mathrm{Bet}(n, k)$.

- Extremwertverteilungen:
  Gumbel–Verteilung: $P(X \leq x) = \exp(-e^{-x})$,
  Fréchet–Verteilung: $P(X \leq x) = \exp(-x^{-\alpha})$, $x > 0$, $\alpha > 0$
  Weibull–Verteilung: $P(X \leq x) = \exp(-(-x)^{\alpha})$, $x < 0$, $\alpha > 0$
- *Pareto–Verteilung:* $P(X \leq x) = 1 - (\frac{b}{x})^a$, $0 < b < x$

2.1.3 (e) Mischungen von Verteilungen

- im Kontext von robuster Statistik wichtig (nicht nur dort!)
- $P^X = (1-r)P^0 + rP^c$, mit $r \in (0,1)$ (Kontaminationsradius).
- $X = (1-U)X^{\text{id}} + UX^{\text{cont}}$, X^{id}, X^{cont}, $U \sim \text{Bern}(r)$ sto. u.

2.1.4 Umsetzung in R

- umfassende Sammlung an Verteilungen bereits in R verfügbar
- Nomenklatur: `<prefix><VtlgName>`, z.B. **dnorm**
  mit `VtlgName`: Name der Verteilung und `<prefix>`: $\in$
  {`r`, `d`, `p`, `q`}

- `r` : Simulation von Zufallsgrößen mit Verteilung `<VtlgName>`
  zusätzliches Argument: `n` $\hat{=}$ Stichprobenumfang
- `d` : Dichte / W-fkt der Verteilung `<VtlgName>`
  zusätzliches Argument: `x` $\hat{=}$ Auswertungsstelle(n)
- `p` : Verteilungsfkt. der Verteilung `<VtlgName>`
  zusätzliches Argument: `q` $\hat{=}$ Auswertungsstelle(n)
- `q` : Quantilsfkt. der Verteilung `<VtlgName>`
  zusätzliches Argument: `p` $\hat{=}$ Auswertungsstelle(n)

- dabei Quantilsfunktion:

$$\begin{aligned} q^X : \quad [0,1] &\to \mathbb{R} \\ s &\mapsto q(s) = \inf\{t \in \mathbb{R} \,|\, F^X(t) \geq s\} \end{aligned} \tag{2.1.1}$$

  speziell
  - $q^X(1/2) \hat{=}$ Median,
  - $q^X(1/4)$, $q^X(3/4) \hat{=}$ unteres/oberes Quartil

Tabelle 2.1-1 [Verteilungen in R]:

— aus http://cran.r-project.org/doc/manuals/R-intro.pdf, p. 34

Verteilung	<VtlgName>	Parameter
Bet	**beta**	shape1,shape2,ncp
Bin	binom	size,prob
Cauchy	cauchy	location,scale
χ^2_m	chisq	df,ncp
Exp	**exp**	rate
$F_{m,n}$	f	df1,df2,ncp
Gam	**gamma**	shape,scale
Geo	geom	prob
HypGeo	hyper	m,n,k
log–$\mathcal{N}$	lnorm	meanlog,sdlog
logist	logis	location,scale
negBin	nbinom	size,prob
$\mathcal{N}$	norm	mean,sd
Poiss	pois	lambda
t_n	**t**	df,ncp
ufo	unif	min,max
Weibull	weibull	shape,scale
Wilcox	wilcox	m,n

Extremwertvtlg.'en in Paket evd

2.2 Verteilungen mit den Zusatzpaketen `distr` und `distrEx`

im Anschluss dieses Kurses:

- Software-Praktikum von *Thomas Stabla*, `statho3@web.de`, *Florian Camphausen*, `fcampi@gmx.de`
- Ergebnis: R–Paket "distr"
- mittlerweile (05/2007) in Version `1.9` auf `CRAN` veröffentlicht ; siehe auch Ruckdeschel et al. (2006).
- erweitert durch Paket "distrEx"

Auslöser für diese Pakete:

- will Algorithmus unabhängig von einer konkreten Verteilung formulieren;

- dafür notwendig: tatsächliche Verteilung als Art Variable an Algorithmus übergeben
- Im Prinzip möglich: zusammenkleben von Präfix und Verteilungsname in **eval**(**parse** (....))
- unelegant und unflexibel
- stattdessen: Variablentyp (genauer Klasse) Distribution
- Arithmetik für Verteilungen:
  - Identifikation von Zufallsvariable und Verteilung
  - **Norm**()+**Pois**() bedeutet:
    $X \sim \mathcal{N}(0,1)$, $Y \sim \mathrm{Pois}(1)$, X, Y sto. u., dann:
    Norm()+**Pois**() $\hat{=}$ $\mathcal{L}(X+Y)$
  - analog **sin**(**Norm**())

R–Beispiel 2.2-1:

```
library(distr)
N ← Norm(mean = 2, sd = 1.3)
P ← Pois(lambda = 1.2)
Z ← 2*N + 3 + P ;  Z
plot(Z)
p(Z)(0.4); q(Z)(0.3)
Zs ← r(Z)(1000); Zs[1:30]
```

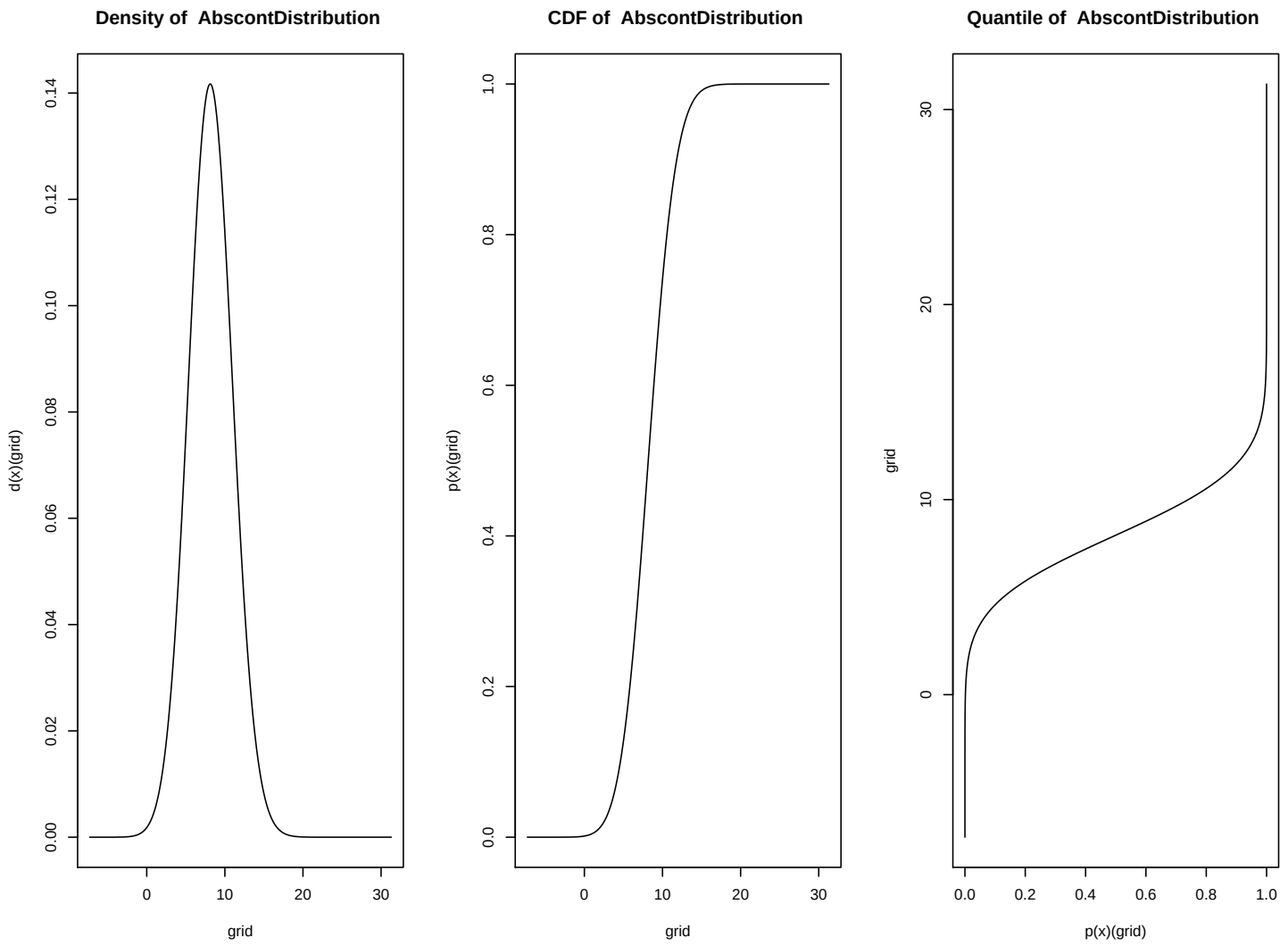



Abbildung 1: Dichte, Verteilungsfunktion un Pseudoinverse von Z

In Paket `distrEx` hat man auch Funktionale auf Verteilungen wie Erwartungswert (E), Varianz (**var**), Standardabweichung (**sd**), etc. zur Verfügung:

R–Beispiel 2.2-2:

```
library(distrEx)
E ← Exp(rate=2)
E(E) ## benutzt Formel =1/rate
E(as(E,"AbscontDistribution")) ## mit num. Integration
E(as(E,"UnivariateDistribution")) ## mit Simulationen
E(E, fun = function(x){2*x^2}) ## berechnet E(2E^2)
# mit demselben Operator/Funktional
P ← Pois(lambda=2)
E(P) ## nutzt Formel =lambda
E(as(P,"DiscreteDistribution")) ## mit Summen
E(as(P,"UnivariateDistribution")) ## mit Simulationen
E(P, fun = function(x){2*x^2}) ## berechnet E(2P^2)
```


2.3 Simulation von Zufallsvariablen

2.3.1 Was sind "gute" Zufallszahlen?

- sollen "zufällig" aussehen
- besser: sollen mit statistischen Verfahren (Tests) nicht von $\mathrm{u.i.v.}$ $\mathrm{ufo}(\{0, \ldots, N-1\})$ unterscheidbar sein, insbesondere
  - von einer bekannten Verteilung
  - keine Autokorrelation
  - keine höheren Abhängigkeitsschemata
- sollen eine lange Periode haben (vor Wiederholung)
- sollen einfach und schnell produzierbar sein
- sollen (durch Setzung eines Initialisierungs–Parameters) reproduzierbar sein

Die folgenden drei Unterabschnitte stellen in einem Exkurs wichtige Resultate und Prinzipien zur Simulation von Zufallsvariablen zusammen; da diese aber sowieso in R implementiert sind, kann man diesen Abschnitt auch gut überspringen.

2.3.2 Schritt 1: Erzeugung von $X \sim \text{ufo}(\{0, \ldots, N\})$; typische Zufallszahlengeneratoren

2.3.2 (a) historisch: physikalisch erzeugte Zufallszahlen

- z.B. C-14–Zerfall, andere Isotopenzerfälle

2.3.2 (b) lineare Kongruenzenschemata

DEFINITION 2.3-1:

Für $0 \leq r_0, a, c \leq m$ definiert man "Zufallszahlen" r_i als

$$r_{i+1} = (ar_i + c) \bmod m \qquad (2.3.1)$$

Hierzu gilt:

THEOREM 2.3-2:
Ein Schema gemäß (2.3.1) hat die (maximale) Periode m genau dann, wenn

(i) $\mathrm{ggT}(c, m) = 1$

(ii) $a \equiv 1 \bmod p$ *für jeden Primfaktor* $p \mid m$

(iii) falls $4 \mid m$, *dann* $a \equiv 1 \bmod 4$

Im Spezialfall $m = 2^e$, $e \geq 2$ *haben wir maximale Periode genau dann, wenn* c *ungerade und* $a \equiv 1 \bmod 4$.

Für den Beweis verwenden wir

LEMMA 2.3-3 [KLEINER FERMAT'SCHER SATZ]:
Für p *prim gilt für* $0 \leq a \leq p - 1$

$$a^p - a \equiv 0 \bmod p. \tag{2.3.2}$$

Beweis zu Lemma 2.3-3:

Für $a \equiv 0 \bmod p$ ist (2.3.2) trivial; sei also $a \not\equiv 0 \bmod p$. Dann ist (2.3.2) äquivalent mit $a^{p-1} \equiv 1 \bmod p$. Aber die Multiplikation mit a stellt einen Gruppenautomorphismus φ in der abelschen, multiplikativen Gruppe $\mathcal{G}$ der Restklassen $[1], \ldots, [p-1]$ dar. Daher gilt

$$[0] \neq \prod_{h \in \mathcal{G}} h = \prod_{h \in \mathcal{G}} \varphi(h) = \prod_{h \in \mathcal{G}} ([a]h) = [a]^{|\mathcal{G}|} \prod_{h \in \mathcal{G}} h$$

also in der Tat $[1] = [a]^{|\mathcal{G}|} = [a]^{p-1}$. ////

Beweis zu Theorem 2.3-2:

"$\Longrightarrow$":
Haben wir volle Periode m so durchlaufen wir in m Schritten alle Restklassen $[0], \ldots, [m-1]$ genau einmal. Wo wir das Zählen anfangen ist dabei irrelevant, ohne Einschränkung also mit $r_0 = [0]$.
Wir fangen mit Bedingung (i) an. Wir verwenden hierfür eine Relation, die wir allgemeiner für $0 < p \leq m$ brauchen können: Mit Induktion sehen wir ein, dass für $r_0 = 0$ gilt, sofern $a \neq 1 \bmod p$

$$r_n \bmod p \;=\; \sum_{i=0}^{n-1} a^i c \bmod p \;=\; c\frac{a^n - 1}{a - 1} \bmod p \qquad (2.3.3)$$

Denn:

$$r_0 = 0 \bmod m = 0 \bmod p = c\frac{a^0-1}{a-1} \bmod p$$

$$r_n \bmod p = ar_{n-1} + c \bmod p \overset{\text{IV}}{=} c + ac\frac{a^{n-1}-1}{a-1} \bmod p = c\frac{a^n-1}{a-1} \bmod p$$

und im Fall $a = 1 \bmod p$ gilt offenbar $r_n = nc \bmod p$.
Damit folgt für $p = m$ dass $c \mid r_i \bmod m$ für alle $i > 0$. Gilt nun $d := \text{ggT}(c, m) > 1$ und $m = m_0 d$, so gilt auch $d | r_i \bmod m$ für alle $i > 0$, und somit durchläuft $r_i \bmod m$ nur Vielfache von d in $0, \ldots, m$ und die maximale Periode ist $m_0 < m$. Also gilt (i).
Wir nehmen nun an $qm_q = m$ für $q, m_q \in \mathbb{N}$ und betrachten die Folge der r_i modulo q. Wir zeigen nun

$$\text{Periode}(r_n \bmod m) = m \quad \Longrightarrow \quad \text{Periode}(r_n \bmod q) = q \tag{2.3.4}$$

Wegen der Restklassenbildung gilt für die Periode λ für diese Folge $\lambda \leq q$; nehmen wir an, $\lambda < q$, so müsste gelten $\lambda \mid q$. dann aber folgte dass die maximale Periode von $r_n \bmod m$ nur $m_q\lambda < m$ sein könnte, ein Widerspruch. Also muss (2.3.4) gelten.
Für (ii) verwenden wir dies mit $q = p$ einem Primteiler von m. Nehmen wir nun an, $a \not\equiv 1 \bmod p$. Dann folgt nach (2.3.3) $0 = c\frac{a^p-1}{a-1} \bmod p$. Dass $c \equiv 0 \bmod p$ wird durch (i) widerlegt, und somit müsste $a^p - 1 \equiv 0 \bmod p$ sein; dies aber stünde im Widerspruch zu Lemma 2.3-3, und so gilt (ii).

Nehmen wir nun an, dass $4 \mid m$. Man betrachte die Folge der r_i modulo 4. Dann aber muss nach (2.3.4) diese Folge Periode 4 haben. Sei $a \not\equiv 1 \bmod p$. Wegen (ii) muss dann $a \equiv 3 \bmod 4$ sein; und wegen (i) muss gelten $c \equiv 1$ oder $3 \bmod 4$. Für diese beiden Konstellationen ergeben sich:

$$a \equiv 3 \quad c \equiv 1 \bmod 4 \qquad \Rightarrow \ (r_i) = (0, 1, 0, 1, \ldots)$$
$$a \equiv 3 \quad c \equiv 3 \bmod 4 \qquad \Rightarrow \ (r_i) = (0, 3, 0, 3, \ldots)$$

also keine volle Periode. [Beweis wird auf Seite 206 fortgesetzt]

Vor dem Beweis der Rückrichtung noch drei Lemmata:

LEMMA 2.3-4:
Sei $m = st$ für $\mathrm{ggT}(s,t) = 1$ und λ_m, λ_s und λ_t die Perioden des Schemas (2.3.1) für die Parameter (r_0, a, c, m), (r_0', a', c', s) und (r_0'', a'', c'', t), wobei die $\cdot'$–Größen modulo s, die $\cdot''$–Größen modulo t aus den Parametern (r_0, a, c, m) hervorgehen.
Dann gilt $\lambda_m = \lambda_s \lambda_t$.

Beweis zu Lemma 2.3-4:

Da $\lambda_s \,|\, \lambda_m$, $\lambda_t \,|\, \lambda_m$ folgt, dass $\lambda_m \geq \mathrm{kgV}(\lambda_s, \lambda_t) =: \tilde{\lambda}$ Da aber $\lambda_s \,|\, \tilde{\lambda}$ folgt, weil λ_s die Periode des $\cdot'$–Schemas ist, $r_n \equiv r_{n+\tilde{\lambda}} \bmod s$, und analog $r_n \equiv r_{n+\tilde{\lambda}} \bmod t$, so dass also auch $r_n \equiv r_{n+\tilde{\lambda}} \bmod st$, also $\lambda_m \leq \tilde{\lambda}$. ////

Lemma 2.3-5:

Die Periode λ des Schemas (2.3.1) mit $c = 1$ und $a > 1$ ist p^e für eine Primzahl p genau dann, wenn

(i) $a \equiv 1 \bmod p$ *für* $p > 2$

(ii) $a \equiv 1 \bmod 4$ *für* $p = 2$

Lemma 2.3-6:

Sei p prim und $e \in \mathbb{N}$. Dann gilt

$$\left(\begin{array}{l} x \equiv 1 \bmod p^e \\ x \not\equiv 1 \bmod p^{e+1} \end{array} \right) \Longrightarrow \left(\begin{array}{l} x^p \equiv 1 \bmod p^{e+1} \\ x^p \not\equiv 1 \bmod p^{e+2} \end{array} \right) \qquad (2.3.5)$$

Beweis zu Lemma 2.3-6:

Sei also $x = 1 + qp^e$ mit $\mathrm{ggT}(q, p) = 1$. Dann gilt

$$x^p = (1 + qp^e)^p = 1 + p^{e+1} \sum_{l=1}^{p} \binom{p}{l} / pq^l p^{e(l-1)} \equiv 1 \bmod p^{e+1},$$

da der Faktor $\binom{p}{l}$ für $1 < l < p$ stets einen Primfaktor p abspaltet und für $l = p$ dieser Faktor von $p^{e(l-1)}$ geliefert wird. Andererseits gilt für $l = 1$:

$$\binom{p}{l} / pq^l p^{e(l-1)} = q^l \nmid p,$$

also $x^p \not\equiv 1 \bmod p^{e+2}$. ////

Beweis zu Lemma 2.3-5:

Das Lemma ist der Spezialfall des Satzes für $1 < a < p^e$, $c = 1$ und $m = p^e$. "$\Longrightarrow$" ist daher bereits durch den entsprechenden Beweisteil des Theorems abgedeckt.
"$\Longleftarrow$": Sei $a = 1 + qp^f$ mit $p^f > 2$, also (i) oder (ii) in der Aussage von Lemma 2.3-5. Sei weiter $\mathrm{ggT}(p, q) = 1$. Dann folgt aus Lemma 2.3-6 — indem wir induktiv in g in (2.3.5) $a^{(p^{g-1})}$ für x einsetzen für alle $g \geq 0$:

$$\left(\begin{array}{l} a \equiv 1 \bmod p^f \\ a \not\equiv 1 \bmod p^{f+1} \end{array} \right) \Longrightarrow \left(\begin{array}{l} x^{p^g} \equiv 1 \bmod p^{f+g} \\ x^{p^g} \not\equiv 1 \bmod p^{f+g+1} \end{array} \right) \qquad (2.3.6)$$

Aus der rechten Seite der Implikation aber folgt

$$\left(\begin{array}{rcl} a^{(p^g)} & \equiv & 1 \mod p^{f+g} \\ a^{(p^g)} & \not\equiv & 1 \mod p^{f+g+1} \end{array}\right) \Longrightarrow \left(\begin{array}{rcl} \frac{a^{(p^g)}-1}{a-1} & \equiv & 0 \mod p^{g} \\ \frac{a^{(p^g)}-1}{a-1} & \not\equiv & 0 \mod p^{g+1} \end{array}\right) \tag{2.3.7}$$

denn aus der linken Seite folgt $a^{(p^g)} = 1 + sp^{f+g}$ für ein $0 \leq s < p^{f+g}$ so dass $s \nmid p^{f+g}$, also $s \nmid p$. Damit aber ist der Bruch auf der rechten Seite gerade

$$\frac{a^{(p^g)}-1}{a-1} = \frac{sp^{f+g}}{qp^f} = \frac{s}{q}p^g$$

Betrachten wir wieder (2.3.3) mit $r_0 = 0$, $m = p^e$, $a > 1$ und $\mathrm{ggT}(c, p) = 1$. Dann gilt für die Periode λ des Schemas

$$\lambda \,|\, n \quad \Longleftrightarrow \quad r_n \equiv 0 \mod p^e \quad \Longleftrightarrow \quad a^n - 1 \equiv 0 \mod p^e$$

Setzen wir $n = p^e$, so folgt mit $g = e$ aus der rechten Seite der Implikation (2.3.7) $a^{(p^e)} - 1 \equiv 0 \mod p^e$, also $\lambda \,|\, p^e$. Andererseits setzen wir $n = p^{e-1}$, so folgt mit $g = e - 1$ aus der rechten Seite der Implikation (2.3.7) $a^{(p^{e-1})} - 1 \not\equiv 0 \mod p^e$, also $\lambda \nmid p^{e-1}$, so dass in der Tat $\lambda = p^e$. ////

Fortsetzung Beweis : "$\Longleftarrow$" in Theorem 2.3-2:

Nach Lemma 2.3-4 genügt es, das Theorem für Primzahlpotenzen zu zeigen. Dabei genügt es zu zeigen, dass die mit $r_0 = 0$ startende Folge maximale

Periode hat. Das Theorem ist richtig für $a = 1$: Dann ist $r_n \equiv nc \bmod m$, und nach (i) folgt aus $nc \equiv 0 \bmod m$ auch $n \equiv 0 \bmod m$, also was genau dann gilt, wenn $\lambda = m$. Für $1 < a < p^e$ wenden wir Lemma 2.3-5 an. ////

- in der Praxis finden meist Schemata aus Definition 2.3-1 mit $c = 0$, also rein multiplikative Schemata, Verwendung
- $r_0 = 0$ ist hier natürlich verboten!
- die maximale Periode ist hier $m - 1$ (s.u.)
- in R: $r_0 =$ **function**(`seed`), $a = 69069$, $c = 0$, $m = 2^{32}$

Hierzu gilt:

THEOREM 2.3-7:
Sei $m = 2^e$, $e \geq 4$. Dann ist die maximale Periode $m/4 = 2^{e-2}$ und wird genau dann erreicht, wenn $a \equiv 3$ oder $5 \bmod 8$ und r_0 ungerade.

BEWEIS:

in Ripley (1987) ////

THEOREM 2.3-8:

Die maximale Periode $m-1$ kann nur erreicht werden, wenn m prim ist; dann teilt die Periode $m-1$ und ist $m-1$ genau dann, wenn a primitive $(m-1)$–te Einheitswurzel ist, i.e. $a \neq 0$, $a^{\frac{m-1}{p}} \not\equiv 1 \bmod m$ für jeden Primfaktor p von $e = m-1$ und e die kleinste Potenz, für die $a^e \equiv 1 \bmod m$.

BEWEIS:

in Ripley (1987) ////

BEMERKUNG 2.3-9:

- die hintersten Bits von r_n sind nicht sehr zufällig!
  Sei dazu $m = 2^e \geq 256$. Dann benimmt sich das letzte Byte stets wie modulo 256, also mit maximaler Periode 64 ....
- schnelle Multiplikation durch `Shift` und `Add`

$\rightsquigarrow$ Generator `RANDU` mit $a = 2^{16} + 3$

$$r_{n+1} = a r_n \bmod 2^{32}$$

Schreibt man diese Rekursion aus, so erhält man

$$r_{n+2} = (2^{32} * 2^{16} + 9) r_n \qquad (2.3.8)$$

$$r_{n+1} = (\quad 2^{16} + 3) r_n \qquad (2.3.9)$$

$$r_n = (\quad 1) r_n \qquad (2.3.10)$$

also $-9r_n + 6r_{n+1} + r_{n+2} = 0 \Longrightarrow$ die Tripel $m^{-1}(r_n, 6r_{n+1}, r_{n+2})$ liegen auf 15 parallelen Hyperebenen durch den Einheitswürfel....

2.3.2 (d) "Exor-Shift-Schema" für bits

DEFINITION 2.3-10:

Für $p, q \in \mathbb{N}$ kleiner als die Wortlänge und Initialisierungswerte b_j $j = -p+1, \ldots, 0$ sei

$$b_i = b_{i-p} \triangle b_{i-(p-q)} \qquad (2.3.11)$$

´

- *in* R*:* $p = 32$, $q = 15$

2.3.2 (e) Kombinationen mehrerer Techniken

Durch Verknüpfung mehrerer Generatoren kommt man in natürlicher Weise auf lineare Differenzengleichungen

DEFINITION 2.3-11:

Für $0 \leq r_{-d+1}, \ldots, r_0, a_1, \ldots, a_d \leq m$ *definiert man "Zufallszahlen"* r_i *als*

$$r_{i+1} = \sum_{j=1}^{d} a_j r_{i-j} + c \bmod m \tag{2.3.12}$$

- Erzeugung von Zufallsbits $\rightsquigarrow$ Shift–Register Generatoren
- maximale Periode für $m = 2$: $2^d - 1$
- Periode für die meisten Initialisierungen: 6.6×10^{14}

2.3.2 (f) in R implementierte Zufallszahlengeneratoren (RNG's)

- **Wichmann-Hill**
  - basiert auf: Vorschlag von Wichmann und Hill
  - Periode: $6.9536e12$
  - Referenz: Wichmann and Hill (1982)
- **Marsaglia-Multicarry**
  - basiert auf: Vorschlag von Marsaglia
  - Technik: Multiplizieren mit Übertrag (vgl. zweite Bemerkung 2.3-9)
  - Periode: $\geq 2^{60}$; laut Marsaglia: alle Tests bestanden
  - Referenz: Marsaglia and Zanan (1994)

- **Super-Duper**
  - basiert auf: Vorschlag von Marsaglia aus den 70ern
  - Technik: Kombination von "Exor-Shift-Schema" und rein–multiplikativem Schema
  - versagt beim MTUPLE test der " Diehard battery "
  - Periode: $\sim 4.6 * 10^{18}$
  - Referenz: Marsaglia (1997)
- **Mersenne-Twister**
  - basiert auf: Vorschlag von Matsumoto und Nishimura
  - zur Zeit default
  - Periode: $2^{19937} - 1$
  - Referenz: Matsumoto and Nishimura (1998)
- **Knuth-TAOCP**
  - basiert auf: Vorschlag von Knuth

- Technik: Fibonacci–Folge mit Differenz, i.e.

$$r_j = (r_{j-100} - r_{j-37}) \bmod 2^{30}$$

- Periode: $\sim 2^{129}$
- Referenz: Knuth (1998)

- **Knuth-TAOCP-2002**
  - wie **Knuth-TAOCP** nur mit anderen Seeds

2.3.2 (g) generelle Literatur

- Hammersley and Handscomb (1964)
- Knuth (1998)
- Ripley (1987)

2.3.3 Qualitätskontrolle für Pseudozufallszahlen

2.3.3 (a) Tests auf "Zufälligkeit"

- im Prinzip jeder Test geeignet, besonders aber
- Korrelationstest: sind Teilstichproben unkorreliert,
  vgl. Abschnitt 5.1.13 [$\hat{=}$ auch: Spektraltest!]
- χ^2–Anpassungstest, vgl. Abschnitt 5.1.11

  $_{1\text{dim}}$ werden alle Intervalle der gleichen Länge gleich häufig belegt?

  $_{\text{dim}>1}$ werden nach Standardisierung auf $[0,1]$ alle
  Teil–(Hyper–)würfel des Einheitsquaders gleich häufig belegt?
- Kolmogoroff–Smirnoff–Test, vgl. Abschnitt 5.1.10
- Gap–Test:
  - erst Standardisierung auf $[0,1]$
  - wähle $0 < a < b < 1$.

  - betrachte Länge k der Folgen $U_j, U_{j+1}, \ldots, U_{j+k}$ so dass keines der Folgeglieder in (a, b) liegt; diese Längen sind zu vergleichen mit der entsprechenden Verteilung der Längen unter dem Modell

- Pokertest
  - erst Standardisierung auf $[0, 1]$
  - partitioniere $[0, 1]$ in fünf Teilintervalle.
  - betrachte die Verteilung, dass jeweils fünf aufeinanderfolgende Zufallszahlen in $1, 2, \ldots, 5$ verschiedene Teilintervalle fallen.
  - die Fenster sollen sich dabei nicht überlappen!
  - diese empirische Verteilung ist zu vergleichen mit der entsprechenden Verteilung unter dem Modell

- Coupon–Collector–Test:
  bestimme für $d = 0$ bis Periode-1, wie lange man warten muss, bis jede Zahl 0 bis d einmal vorgekommen ist; diese Wartezeiten

sind zu vergleichen mit der entsprechenden Verteilung unter dem Modell

- Run–Test
  zähle die Längen monoton steigender Teilstücke und vergleiche die Verteilung mit der entsprechenden Verteilung unter dem Modell
- Maximum–Test:
  - erst Standardisierung auf $[0, 1]$
  - bilde nichtüberlappende Gruppen der Länge k
  - Betrachte die Verteilung von
    $Y_j = \max(X_{lk}, X_{lk+1}, \ldots, X_{(l+1)k-1})$
  - Diese empirische Verteilung ist zu vergleichen mit der entsprechenden Verteilung unter dem Modell

2.3.3 (b) Wie zufällig müssen (simulierte) ZV's sein?

- je nach Zweck unter Umständen gar nicht "u.i.v."–Variablen nötig: MC–Integration
- zum Erzeugen "plausibler" Werte auch andere Techniken möglich

2.3.4 Schritt 2: Anamorphose

— Wandlung in "beliebige" Verteilungen

2.3.4 (a) Wandlung in $\mathrm{ufo}([0,1])$

- erst Casten in `float`/`numeric`
- dann durch N teilen

2.3.4 (b) Quantilsfunktion

- Vorbereitung:
  - betrachte die Pseudoinverse einer reellwertigen Zufallsvariable

$$q^X(s) = (F^X)^{-1}(s) = \inf\{t \in \mathbb{R} \,|\, F^X(t) \geq s\} \tag{2.3.13}$$

  - Es gilt die Gleichheit der beiden Mengen für jedes $s \in [0,1]$:

$$\{t \in \mathbb{R} \,|\, (F^X)^{-1}(s) \leq t\} = \{t \in \mathbb{R} \,|\, s \leq (F^X)(t)\}$$

  - Daher gilt für $U \sim \mathrm{ufo}([0,1])$

$$\{\omega \,|(F^X)^{-1}(U(\omega)) \leq t\} = \{\omega \,|U(\omega) \leq (F^X)(t)\}$$

    also $P((F^X)^{-1}(U) \leq t) = P(U \leq F^X(t)) = F(t)$

- also $q^X(U) = (F^X)^{-1}(U) \sim F^X$
- Beispiele:
  - $F^X = \mathrm{Cauchy} \Longrightarrow (F^X)^{-1}(s) = \tan(\pi(s - 1/2))$
  - $F^X = \mathrm{Exp}(\lambda) \Longrightarrow (F^X)^{-1}(s) = -\log(1-s)/\lambda$,
    und da $\mathcal{L}(1 - \mathrm{ufo}[0,1]) = \mathrm{ufo}[0,1]$ wenden wir $-\log(s)/\lambda$ auf die ufo–Variablen an

– $F^X = \mathrm{Bin}(n,p)$: simuliere n u.i.v. $\mathrm{ufo}[0,1]$–Variablen Y_j und bilde $X = \sum_{j=1}^{n} \mathrm{I}_{\{Y_j \geq p\}}$

2.3.4 (c) Rejection Sampling

- Situation: f, g Lebesgue–Dichten von W-Maßen.
- $\exists c > 0$: $f(x) \leq cg(x)$ für alle x
- die Erzeugung von Zufallszahlen gemäß $G(dx) = g(x)\lambda(dx)$ ist leicht zu realisieren

ALGORITHMUS 2.3-12 [REJECTION SAMPLING]:

(1) Generiere $Y \sim G$.

(2) Generiere $U \sim \mathrm{ufo}(0,1)$.

(3) Falls $U \leq f(Y)/(cg(Y))$, setze $X = Y$ sonst gehe zu (1)

THEOREM 2.3-13:

(i) $X \sim F$.

(ii) *Die Ws., ein* X *anzunehmen, ist* $1/c$.

(iii) *Die Zahl* N *der Versuche, bis ein* X *anzunehmen ist, ist gemäß* $\mathrm{Geom}(1/c)$ *verteilt; insbesondere gilt*

$$\mathrm{E}[N] = c, \qquad \mathrm{Var}[N] = c^2 - c \tag{2.3.14}$$

BEWEIS:

$$P(X \le t) = P(Y \le t \,|\, U \le f(Y)/(cg(Y))) = \frac{P(Y \le t,\, U \le \frac{f(Y)}{cg(Y)})}{P(U \le \frac{f(Y)}{cg(Y)})}$$

Für den Zähler gilt

$$P(Y \leq t,\, U \leq \frac{f(Y)}{cg(Y)}) = \int \mathrm{I}_{\{y \leq t,\, u \leq \frac{f(y)}{cg(y)}\}}\, \mathrm{I}_{u \in [0,1]}\, \lambda(du)\, G(dy) =$$

$$= \int_{(-\infty;t]} \int_0^1 \mathrm{I}_{\{u \leq \frac{f(y)}{cg(y)}\}}\, \lambda(du)\, G(dy) = \int_{(-\infty;t]} \frac{f(y)}{cg(y)}\, G(dy) =$$

$$= \int_{(-\infty;t]} \frac{f(y)}{cg(y)}\, g(y)\lambda(dy) = \frac{1}{c} \int_{(-\infty;t]} f(y)\lambda(dy) = \frac{1}{c} F(t)$$

und analog für den Nenner

$$P(U \leq \frac{f(Y)}{cg(Y)}) = \frac{1}{c} \int_{(-\infty;\infty)} f(y)\lambda(dy) = \frac{1}{c}$$

und so $P(X \leq t) = F(t)$, also (i).
$P(\text{“Annahme von } X\text{”}) = P(U \leq \frac{f(Y)}{cg(Y)}) = 1/c$, der Rest ist klar. ////

- Beispiel: Simulation von $\mathrm{Gamma}(n, \lambda)$–verteilten Größen
  - Situation: n recht groß $\rightsquigarrow$ Simulation als Summe von n unabhängigen $\mathrm{Exp}(\lambda)$–Variablen zu "teuer"
  - Dichte f von $\mathrm{Gamma}(n, \lambda)$:
    $f(x) = \lambda \, \mathrm{I}_{\{x \geq 0\}} \exp(-\lambda x)(\lambda x)^n / n!$
  - Idee: $f(x) \leq c\lambda \exp(-\lambda/n)$, also $G = \mathrm{Exp}(\lambda/n)$
  - man erhält als optimales
    $c = [(n-1)!]^{-1} (\frac{n}{(1-1/n)})^n \exp(-n)$, d.h. $\mathrm{E}[N] \approx \frac{e}{\sqrt{2\pi}} \sqrt{n}$

2.3.4 (d) Ausnutzung der Definition / ZGWS

- $\mathcal{N}(\mu, \sigma^2)$: als Verteilung von $\sigma Y + \mu$ für $Y \sim \mathcal{N}(0, 1)$
- χ^2_n–Verteilung: Summe der Quadrate von n u.i.v. $\mathcal{N}(0, 1)$–Variablen
- $F_{n,m}$–Verteilung:
  im wesentlichen als Quotient zweier entsprechender χ^2–Verteilungen
- t_n–Verteilung:
  im wesentlichen als Quotient einer $\mathcal{N}(0, 1)$– und einer entspr. χ^2–Verteilung
- früher (schlecht!): $\mathcal{N}(6, 1)$
  - als Verteilung der Summe von 12 unabhängigen $\mathrm{ufo}[0, 1]$ Variablen
  - erste beide Momente passen $\quad \rightsquigarrow$ Hoffnung auf ZGWS?
- multivariate Normalverteilung $\mathcal{N}_p(\mu, \Sigma)$
  - zuerst jeweils p "unabhängige" $\mathcal{N}(0, 1)$–Variablen als einen $\mathcal{N}_p(0, \mathbb{I}_p)$–verteilten Vektor Y auffassen
  - $\mathcal{N}_p(\mu, \Sigma)$ als Verteilung von $SY + \mu$, wobei S beliebig mit $SS^\tau = \Sigma$, z.B. "Choleski-Hälfte"

2.3.4 (e) Box–Muller für Normalverteilung

- betrachte für $Y \sim \mathcal{N}_2(0, \mathbb{I}_2)$ Y im wesentlichen in Polarkoordinaten, i.e.

$$(r(Y)^2, \rho(Y)) = (\|Y\|^2/2, \arctan(Y_2/Y_1)) \tag{2.3.15}$$

- wegen der Rotationsinvarianz von $Y \sim \mathcal{N}_2(0, \mathbb{I}_2)$ unter $O \in \mathrm{Orth}(2)$ ist $\rho(Y)$ gleichverteilt auf allen Winkeln
- $\|Y\|^2 \sim \chi^2_2$, also $\rho(Y) \sim \mathrm{Exp}(1)$
- dies liefert:

ALGORITHMUS 2.3-14 [BOX–MULLER]:

(1) Generiere $U_1, U_2 \overset{\text{u.i.v.}}{\sim} \mathrm{ufo}(0,1)$.

(2) $X = 2\pi U_1$

(3) $Z = -\log(U_2)$

(4) $Y_1 = \sin(X)\sqrt{2 * Z}, \qquad Y_2 = \cos(X)\sqrt{2 * Z}$

2.3.4 (f) Wartezeiten bei Poissonvariablen

- Ziel: $\mathrm{Poisson}(\lambda)$–verteilte Zufallsvariablen
- nutze aus:

  THEOREM 2.3-15:
  Seien $L_i \overset{\text{u.i.v.}}{\sim} \mathrm{Exp}(\lambda)$, $T_k := \sum_{i=1}^{k} L_i$. *Dann ist der Prozess* $\{N_t\}$, $N_t := \sum_{k=1}^{\infty} \mathrm{I}_{\{T_k \le t\}}$ *ein Poissonprozess mit konstanter Intensität* λ.

  BEWEIS:
  Georgii (2002), Satz 3.33 ////

- Definiere $N_\lambda := \min\{k \ge 1 : T_k > \lambda\}$. Dieses ist nach Theorem 2.3-15 $\mathrm{Poisson}(\lambda)$–verteilt

- dies ergibt

  ALGORITHMUS 2.3-16:

  (0) $v := 1$; $k := -1$;

  (1) Generiere $U \sim \text{ufo}(0, 1)$.

  (2) $v := Uv$; $k := k + 1$;

  (3) falls $v < e^{-\lambda}$ gehe zu (1) sonst

  (4) $N_\lambda := k$

- Algorithmus 2.3-16 ist gut für λ klein
- für λ groß besser geeignete Quantilstransformation

- ACHTUNG: bei Verwendung der Exp-Verteilung deren Gedächtnislosigkeit berücksichtigen;
- Genauer gilt für X $\mathrm{Exp}(\lambda)$ und $s, t > 0$ —
  (und unter stetigen Verteilungen nur für $\mathrm{Exp}(\lambda)$):

$$P(X > t + s | X > s) = P(X > t) \tag{2.3.16}$$

- der Parameter $\lambda \mathrel{\hat{=}}$ mittlere Zahl an Ereignissen in vorgegebener Zeit

2.3.4 (g) Wartezeiten bei Poissonvariablen und nicht-konstanter Intensität

- gegeben: zeitvariable Intensitätsfunktion $\lambda(t)$, $\lambda(t) \geq 0$:
- Übergang zur Stammfunktion $\Lambda(t) = \int_{T_0}^{t} \lambda(s)\, ds$
  $\Longrightarrow$ Λ isoton — in R:

```
stammfunktion<-function(fun,T0=0,T1,nrgitter=1000){#
x<-seq(T0,T1,length=nrgitter)
Lf<-function(t){# berechnet g(t) an der Stelle t
    return(integrate(fun,T0,t,rel.tol=10^-8)$value)}
y<-sapply(x,Lf)
return(splinefun(x,y))
}
```

- Bildung der Pseudoinversen $\Lambda^-(x) = \inf\{s \geq T_0 \,|\, \Lambda(s) \geq x\}$ — in R:

```
invertiere<-function(fun,T0=0,T1,nrgitter=1000){#
x<-seq(T0,T1,length=nrgitter)
y<-fun(x)
return(splinefun(y,x))
}
```


- um also Λ^- im (y-)Bereich $\texttt{t0} = 8/24$; $\texttt{t1} = 18/24$ ($\hat{=}$ein Werktag) zur gegebenen R-Funktion **lambda** zu bekommen, geht man nun so vor:

```
Lbd1 ← invertiere (stammfunktion(lambda,T0=t0,T1=t1),T0=t0,T1=t1)
```

- Tatsache: folgende Vorgehensweise liefert Wartezeiten zur gegebenen Intensitätsfunktion **lambda**

```
# 1 # Ziehe N (N gross) Exp(1)-Variablen:
    # roh-Zwischenankunftszeiten
Ex1 ← rexp(N)
# 2 # gehe über zu kumulativen Summen:
    # roh-Ankunftszeiten
CEx1 ← cumsum(Ex1)
# 3 # Transformation mit Lambda^-
    # Ankunftszeiten
Ankunftszeiten ← Lbd1(CEx1)
```


Schritte bei empirischer Bestimmung von $\lambda(t)$:

- Erstellung / Erhebung einer Stichprobe von Ankunftszeiten hier: `Ankunft`
- Aufgabe: Rekonstruktion von λ aus diesen Daten
- erst: Ermittlung der Wartezeiten und der empirischen Raten:

```
### Umrechnung von Wartezeiten in empirische Raten
##Wartezeiten
WZ=diff(Ankunft)
WZs=as.numeric(floor(WZ*60*24*60))
## Wartezeit 0 verhindern
WZs[WZ==0]=runif(WZs==0)
rate=1/WZs
```


- dann: Glättung

```
## starke Glättung (spar nahe bei 1)
erg <- smooth.spline(x=Ankunft, y=rate, spar=0.9,
                     all.knots=TRUE)
plot(erg, type="b")
##Glättungsfunktion
sf <- splinefun(erg$x, erg$y)
##interessierender Zeitbereich (in "Tagen")
x <- seq(8*60,18*60,length=200)
xt <- times(seq(8/24,18/24,length=200))
sfy <- sf(x); sfy <- sfy*(sfy>0)
plot(xt, sfy, type="l")
points(Ankunft,0*Ankunft, pch="*")
points(Ankunft, rate, pch="+")
```


2.4 Univariate, num. Kenngrößen

2.4.1 die empirische Verteilung

2.4.1 (a) Definition

- Situation: haben reale / simulierte Daten aus "unbekannter" Verteilung
- relevante Information bereits in empirischer Verteilung
- definiert als

$$F^{\text{emp}}(t) := \frac{1}{n} \sum_{i=1}^{n} \mathrm{I}_{\{X_i \leq t\}}$$

2.4.1 (b) Umsetzung in R

- Funktion ecdf
- siehe auch Abschnitt 2.5.2
- empirische Quantile mit **quantile**

2.4.2 Zusammefassungen

2.4.2 (a) der Befehl summary

- gibt Mittelwert, Quartile und die Zahl der Missings (falls vorhanden) aus

2.4.2 (b) Fivenum

- definiert als Vektor aus Minimum, unterem Quartil, Median, oberem Quartil und Maximum
- in R als **quantile**(x) — per default falls kein Quantil–Argument

2.4.2 (c) Stem and Leaf

- "Text–Version" eines Histogramms (c.f. Abschnitt 2.5.1)
- Daten in Zeilen nach führenden Ziffern aufgeteilt
- in R als **stem**(x)

2.4.3 Lokationsmaße

- Problem: "Wo liegen die Daten / ihr 'Schwerpunkt' ?"
- klassische Lösung: Mittelwert — in R **mean**
- robuste Lösung: Median — in R **median**
- für kategorielle Merkmale: Modus — in R
  ((1:max(x))[rev(order(tabulate(x)))])[1]
  siehe auch http://wiki.r-project.org/rwiki/doku.php?id=tips:stats-basic:modalvalue

2.4.4 Streuungs–/Dispersionsmaße

- Problem: "Wie stark variieren die Daten?"
- klassische Lösung: Varianz und Standardabweichung — in R **var** und **sd**
- robuste Lösung: MAD und IQR — in R **mad** und **IQR**
- extreme Lösung: die Spannweite — in R **max**(x)−**min**(x)
- für kategorielle Merkmale: Shannon–Entropie —

2.4.5 Symmetrie / Krümmung

- Problem: "Sind die Daten schiefverteilt / steiler als die N.V.?"
- Schiefe-Parameter: "Skewness" —http://en.wikipedia.org/wiki/Skewness— in R **mean**((x−**mean**(x))^3)/**sd**(x)^3 oder skewness (in Paket fpBasics
- Exzess/Kurtosis —http://en.wikipedia.org/wiki/Kurtosis in R **mean**((x−**mean**(x))^4)/**sd**(x)^4-3 oder kurtosis (in Paket fpBasics

2.4.6 Zusammenhangsmaße

- Problem: "Hängt die Annahme von Wert x durch X mit der Annahme von Wert y durch Y zusammen?"
- klassisch: Korrelationskoeffizient — in R **cor**
- für ordinale Merkmale: Rangkorrelationen von Spearman und Kendall — in R in der Bibliothek **c**. test als Listenelement im Ergebnis von **cor**. test

2.4.7 getrimmte und winsorisierte Varianten

- oft zur Robustifizierung klassischer Statistiken verwendet
- anstelle des vollen Datensatzes Berechnung nur auf den "zentralen Werten"
- getrimmte Größen: Berechnung auf Basis der entsprechend verkleinerten Stichprobe — in R durch Übergabe des Parameters trim
- winsorisierte Größen: Werte größer als der p% –größte Wert werden (intern) auf diesen gesetzt (entsprechend bei kleinen Werten)

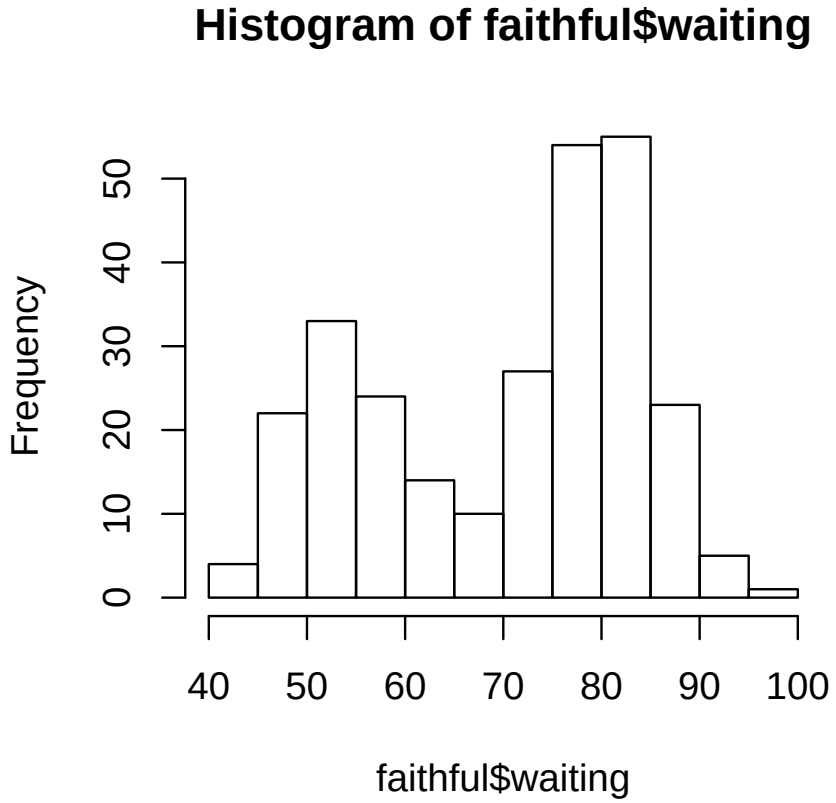
- Beispieldatensatz: x←**c** (100,1,4,3,3,2,−30,3,1,0)
  - Mittelwert **mean**(x) ergibt 8.7
  - 10%-getrimmter Mittelwert **mean**(x,trim=0.1) ergibt 2.125 — durch Anwendung auf $(1, 4, 3, 3, 2, 3, 1, 0)$
  - 10%-winsorisierter Mittelwert
    x1←**sort**(x); **mean**(x1[2:9],x1 [2], x1 [9]) ergibt 2.1
    — durch Anwendung auf $(4, 1, 4, 3, 3, 2, 0, 3, 1, 0)$

2.5 graphische univariate Analyse

Beispiel: der Datensatz `faithful` aus der Bibliothek `MASS`

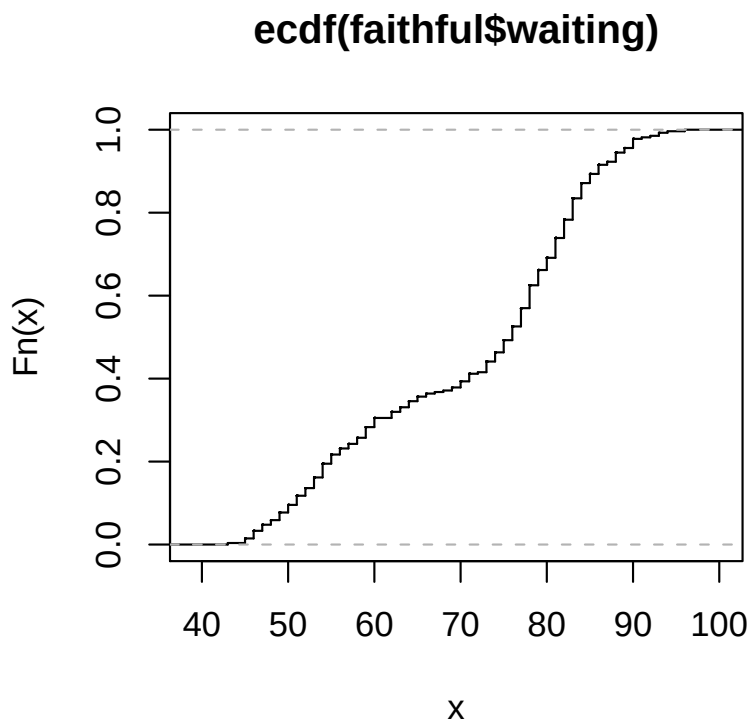
2.5.1 Histogramm

R–Beispiel 2.5-1 [Wartezeiten des Old Faithful]:



- Umsetzung in R: **hist**
- Parameter:
  - Klassenzahl `nclass` — (sollte $\sim \text{const } n^{-1/3}$ sein)
  - `freq` (logisch): relative oder absolute Häufigkeiten
  - `col`: Nummer der Farbe; per default `NULL`, dann keine Füllung

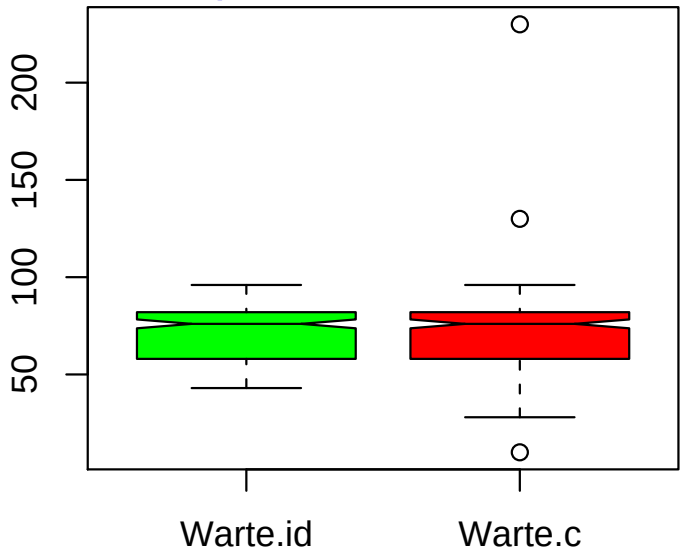
2.5.2 empirische Verteilungsfunktion



- c.f. Abschnitt 2.4.1
- Umsetzung in R:

```
library(stepfun)
plot(ecdf(faithful$waiting),do.points=F,
     verticals=T)
```

2.5.3 Boxplots



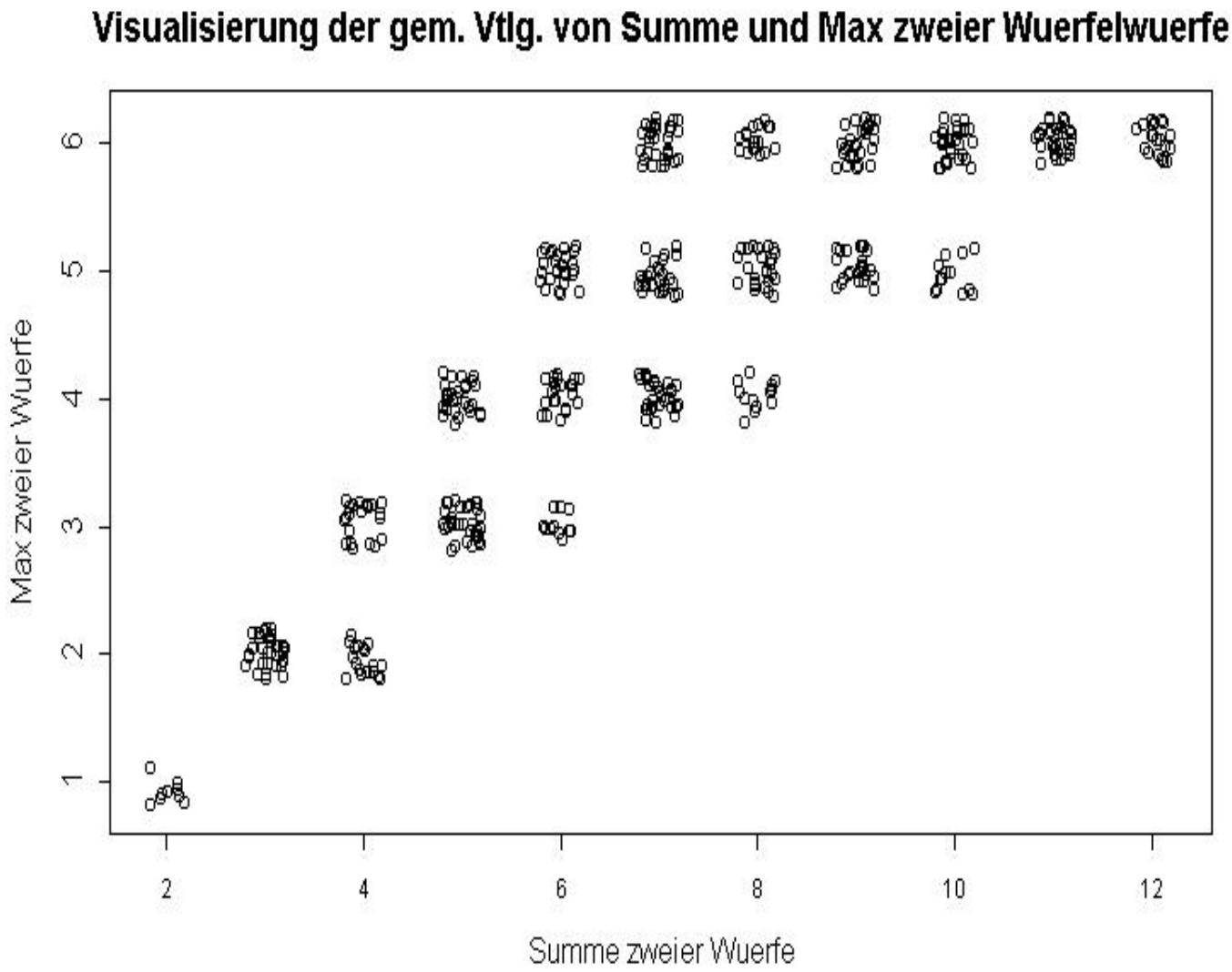
- hier: links Originalwartezeiten, rechts um Ausreißer $(10, 230, 130, 30, 28)$ erweitert
- plottet eine Box zwischen unterem und oberem Quartil mit einer Kerbe am Median
- Striche bis zu den äußersten Beobachtungen innerhalb $\mathrm{med}(x) \pm 1.5\,\mathrm{IQR}(x)$
- Beobachtungen außerhalb der Striche als Kringel $\rightsquigarrow$ Ausreißer
- Umsetzung in R: **boxplot**

2.5.4 Visualisierung diskreter Zufallsvariablen

- Ziel: Visualisierung der einzelnen Realisationen einer diskreten Zufallsvariable X — zum Beispiel
  `X<-sample(1:6,size=50,replace=T)`
- Ideen
  - `plot(X)` — aber hier nicht nötige Information der Ziehungsreihenfolge mit berücksichtigt
  - nur Realisationen: `plot(X,0*X)` — aber hier nur noch die Tatsache sichtbar, dass X den Wert x angenommen hat, nicht wie "oft"
- Ausweg: *jittern*
  d.h. man verwackelt die Daten durch zufällige Störungen, ohne dass aber dabei diese in eine andere Wertausprägung fallen
- Beispiel: `plot(jitter(X),0*X)`
- so auch simultane Betrachtung zweier Variablen möglich

- $\rightsquigarrow$ Visualisierung von Unabhängigkeit/Unkorreliertheit
- Beispiel:

```
X←sample(1:6,size=500,replace=T)
Y←sample(1:6,size=500,replace=T)
S←X+Y
M←pmax(X,Y)
plot(jitter(S),jitter(M))
```

Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

2.6 ein ausgearbeitetes Beispiel

c.f. Venables and Ripley (1999), pp. 121–147 Link aufs File

```
#-*- R -*-

# Sitzung am 27.11.02

# von Venables / Ripley


# Setzen einiger Optionen
# und laden der MASS-library

library(MASS)
options(width=65, digits=5, height=9999)
#
```


```
# 2.1  Statistik / Stochastik

x ← rt(250, 9)
    # erzeugt 250 t_9 Variablen
qqnorm(x); qqline(x)
    # Vergleich mit der NV

# 2.2 ZV's

contam ← rnorm( 100, 0,
                (1 + 2*rbinom(100, 1, 0.05)) )
# knapper geht's nicht --
# Erzeugung von 100 Variablen contam
# contam~ [.95 N(0,1)+.05 N(0,9)]

# 2.3 univariate numerische Kenngroessen und
```


```
# 2.4 graphische univariate Analyse

#
#einige Datensaetze laden

data(geyser)
data(chem)
data(abbey)
help(geyser)
help(chem)
help(abbey)
#
#Plotten einiger Histogramme
# mit unterschiedlicher BW-Wahl
#
par(mfrow=c(2,2))
      # 2 x 2 Bilder pro Seite
```


```
#
hist.scott(geyser$duration, xlab="duration")
hist.scott(chem)
hist.FD(geyser$duration, xlab="duration")
hist.FD(chem)
par(mfrow=c(1,1))
      # 1 Bild pro Seite
#
#weiteren Datensatz laden
#
data(swiss)
help(swiss)
swiss.fertility ← swiss[, 1]

stem(swiss.fertility)
  #stem and leaf plot
stem(chem)
```


```
stem(abbey)
stem(abbey, scale=0.4) # different in R

par(mfrow=c(1,2))
boxplot(chem, sub="chem", range=0.5)
boxplot(abbey, sub="abbey")
par(mfrow=c(1,1))

#

# Robuste Statistiken

sort(chem)
mean(chem)
median(chem)
mad(chem)
unlist(huber(chem))
```


```
unlist(hubers(chem))

sort(abbey)
mean(abbey)
median(abbey)
unlist(hubers(abbey))
unlist(hubers(abbey, k=2))
unlist(hubers(abbey, k=1))
```


2.7 Dichteschätzung

- Besitzt eine Verteilung eine Dichte, so kann man versuchen, diese zu schätzen.
- erster Ansatz: Histogramm
- für Gütekriterium *MSE* (mittlerer quadratischer Fehler) wesentlich bessere Verfahren möglich
- Zerlegung des Kriteriums:

$$\mathrm{MSE} = \mathrm{Var} + \mathrm{Bias}^2$$

- zusätzliche Säulen: Reduktion des Bias aber Erhöhung der Varianz $\rightsquigarrow$ Trade–off
- Literatur: Silverman (1986), Härdle, W. (1991a), Härdle et al. (1998),

2.7.1 Häufigkeitspolygon

- bereits besser, weil glatter: Mittelpunkte der Säulen durch Polygonzug verbinden
- in R: in Bibliothek MASS, Funktion **frequency.polygon**
- R–Beispiel 2.7-1 [Venables, Ripley (1999)]:

```
library(MASS)
data(faithful)
attach(faithful)
hist(eruptions)
frequency.polygon(eruptions)
```

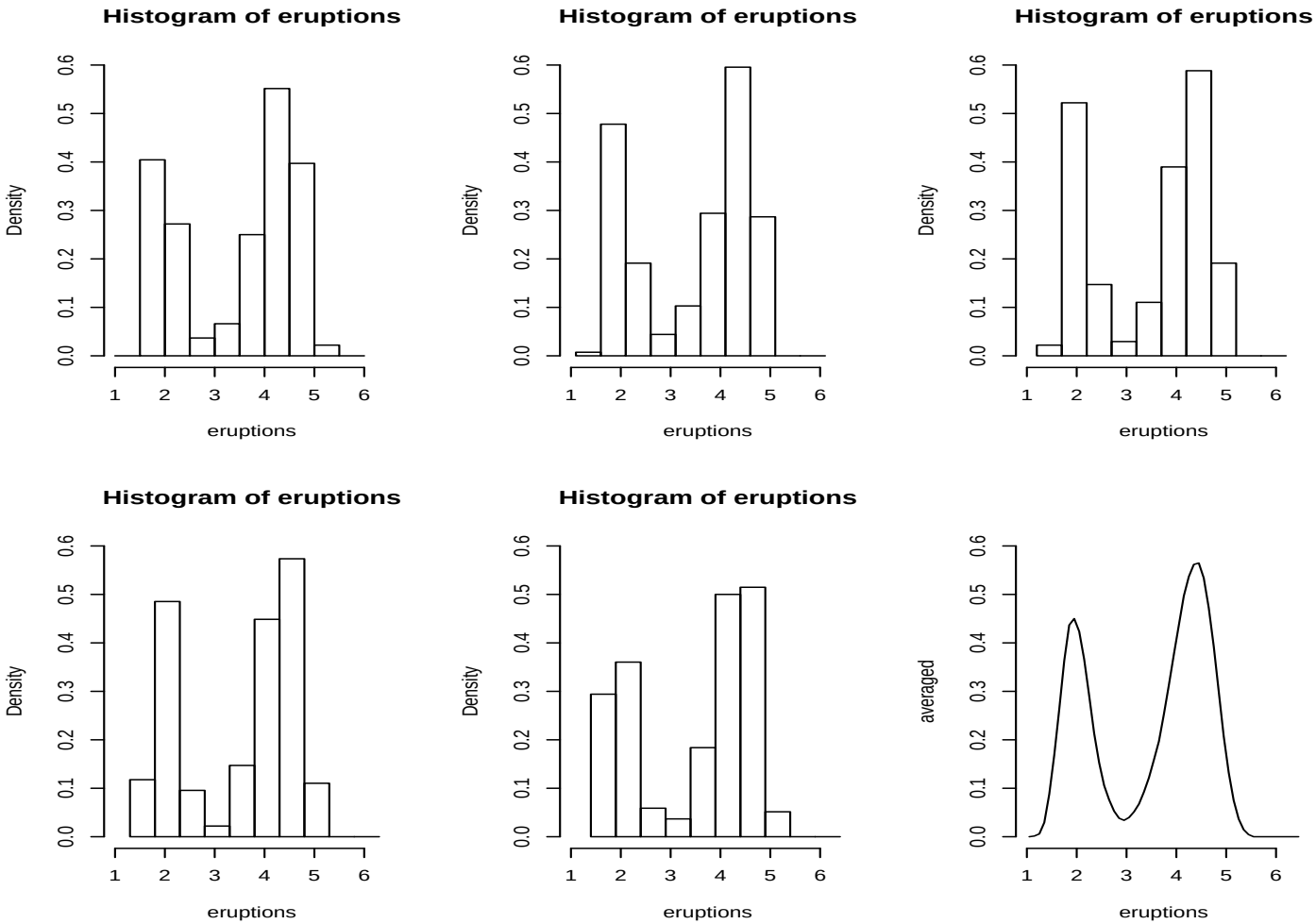

2.7.2 ASH und WARP

- Histogramm und Häufigkeitspolygon hängen vom "Aufpunkt" ab
- $\rightsquigarrow$ mitteln über Aufpunkte: *ASH* (average shifted histogram)
- siehe auch session5.r
- Formel: für Häufigkeiten ν_k

$$\hat{f}(x) := \frac{1}{nh} \sum_{i=1-m}^{m-1} \left[1 - \frac{|i|}{m} \nu_{k+i} \right]$$

- Spezialfall von: *WARP*ing: Weighted Averaging of Rounded Points, Härdle, W. (1991b)

Histogramme zu Geysir Faithful mit versch. Aufpunkten und ASH



Peter Ruckdeschel

Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

2.7.3 Kerndichteschätzung

2.7.3 (a) Idee

- statt über Treppenfunktionen zu mitteln, bereits lokal glätten
- mit einem Glättungskern K und Bandweite h

$$\hat{f}(x) = \frac{1}{h} \sum_{j=1}^{n} K(\frac{x - X_i}{h})$$

- Bandweite steuert Lokalität der Glättung, $\hat{=}$ Säulenzahl bei Histogramm

2.7.3 (b) Eigenschaften von K:

- K ist glatt
- $K \geq 0$

- $\int K(x)\,dx = 1$
- $K(x) = K(-x)$
- am besten $\int |x|^j K(x)\,dx < \infty$ für alle j

2.7.3 (c) verschiedene Kerne

- [Uniform] $K(x) = \frac{1}{2}\,\mathrm{I}_{\{|x|\leq 1\}}$
- [Dreieck] $K(x) = (1-|x|)\,\mathrm{I}_{\{|x|\leq 1\}}$
- [Epanechnikow] $K(x) = \frac{3}{4}(1-x^2)\,\mathrm{I}_{\{|x|\leq 1\}}$
- [Quartik] $K(x) = \frac{15}{16}(1-x^2)^2\,\mathrm{I}_{\{|x|\leq 1\}}$
- [Triweight] $K(x) = \frac{35}{32}(1-x^2)^3\,\mathrm{I}_{\{|x|\leq 1\}}$
- [Gauß] $K(x) = \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)$
- [Cosinus] $K(x) = \frac{\pi}{4}\cos(\pi 2x)\,\mathrm{I}_{\{|x|\leq 1\}}$

2.7.3 (d) Bandweitenwahl

- Kriterium: *AMISE*: approximate mean integrated squared error; Näherungsformel für $\int |f(x) - \hat{f}(x)|^2\,dx$

- generell $h \propto n^{-1/5}$, AMISE bei glattem K : $\propto n^{-4/5}$
- Vergleich: Histogramm AMISE $\propto n^{-2/3}$, im parametrischen Kontext (LLN) $\mathrm{MSE} \propto n^{-1}$
- Verschiedene Möglichkeiten:
  - Rule of Thumb
  - Cross Validation bcv, ucv
  - asymptotische Entwicklung des MISE width.SJ
  - Plug–in –Verfahren
  - Literatur: Härdle et al. (1998), Abschnitt 3.2

2.8 Anwendungen von Zufallszahlen

2.8.1 Simulation

- Situationen
  - aus Sicht des Praktikers
    - reale Datensätze zu klein
    - Vergangenheitsdaten repräsentativ für Zukunft? z.B. Crash–Simulationen in Banken
    - Verteilungsannahmen bei realen Daten fraglich
    - Beispiel: Wechselkurs–Risiken einer Bank

      Man modelliert den Wechselkurs Dollar:Euro zu Zeitpunkt t, W_t, in etwa so:

$$W_t = W_{t-1}(1 + U_t), \qquad U_t \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0, \sigma^2)$$

      Dies macht man auch für weitere Währungen und in ähnlicher Form für den Zins. Um nun den Fremdwährungsbestand nach dem

Niederstwertprinzip in die Bilanz einzustellen, simuliert man viele verschiedene Szenarien, i.e. Wechselkursverläufe im nächsten Jahr, und setzt zum Beispiel das untere 10%–Quantil an.

– aus theoretischer Sicht
  * oft: analytische/theoretische Form der Verteilung schwer / überhaupt nicht zugänglich
  * oder: theoretische Form der Verteilung nur asymptotisch greifbar — wie gut ist die Näherung für finite Stichprobengröße?
  * Beispiel zu ersterem:
    Wie wahrscheinlich nehme ich bei einer Realisation einer Irrfahrt der Länge $n = 100$ mit Start in "0" den Wert "3" zweimal an, bevor ich den Wert "-1" annehme — wissend dass ich die "3" überhaupt zweimal und die "-1" einmal erreiche?
    Antwort mithilfe R–Skript aus File "session6.r": ca. 22%

    - Beispiel zu letzterem:
      Verteilung der Nullstelle von $\sum_{i=1}^{n} \min(k, \max(X_i, -k))$
      für $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0,1)$

- Ausweg
  - $\rightsquigarrow$ Pseudo–Daten mithilfe Pseudo–Zufallszahlen
  - mit Programmen wie R möglich: sehr viele Zufallszahlen sehr schnell erzeugbar
  - Zusammenspiel mit theoretischen Resultaten der Stochastik:
    - Gesetze der großen Zahlen
    - Zentrale Grenzwertsätze
    - Glivenko–Cantelli
  - dadurch: mit hinreichend großer Zahl an Versuchswiederholungen im Prinzip (Achtung: nur Pseudo–Zufallszahlen) beliebige Genauigkeit der Aussagen erreichbar

2.8.2 Daten–Augmentation

siehe gesonderter Abschnitt 2.9

2.8.3 Integration

- zentrales Problem in der Stochastik/Statistik
  — c.f. allgemeiner Integralbegriff, Erwartungswert, Varianz
- wichtige Anwendung auch in der Numerik / Lösung von Differentialgleichungen
- Problemstellung:
  - gegeben eine Funktion $f : \mathbb{R}^k \to \mathbb{R}$, ein Maß P auf $\mathbb{B}^k$ und eine Menge $A \in \mathbb{B}^k$
  - gesucht $\int_A f(x)\, P(dx)$

- numerische Strategie:
  - Approximation von f durch Funktionen, deren Integral analytisch berechenbar (Polynome)
  - dazu: Interpolation von f durch Vorgabe von Stellen x_i
  - für niedrige Dimensionen kaum zu schlagen
  - Problem: in hohen Dimensionen viel zu viele Stützstellen
- stochastischer Ausweg:
  - Ziehung der Stützstellen zufällig
  - Schätzung des Integrals/Erwartungswertes durch Auswertung am empirischen Maß $\rightsquigarrow$ also durch das arithmetische Mittel
  - Gesetze der großen Zahlen $\Longrightarrow$ beliebig genau für hinreichend große Zahl n an Auswertungsstellen
  - Paradigma: Die Genauigkeit wächst mit der Rate $1/\sqrt{n}$ unabhängig von der Dimension !!

Techniken zur stochastischen Integration

— vgl. Ripley (1987), chapter 5

- generelles Problem: [wie oben] gesucht $I = \int_A f(x)\, P(dx)$
- ein einfaches Beispielproblem:
  $X \sim \text{Cauchy}$; gesucht $p = P(X > 2) = \int_2^\infty g(x)\, \lambda(dx)$ mit $g(x) = \frac{1}{\pi} \frac{1}{1+x^2}$
- "Brute–Force": *crude Monte Carlo*
  - simuliere n u.i.v. ZV's X_i gemäß P und schätze I durch $I_n = \text{mean}_i(f(X_i))$
  - im Beispiel: X←**rcauchy**(n); p1←**mean**((X>2))
  - hier: $Y_i = \text{I}_{\{X_i > 2\}} \sim \text{Bin}(1, p)$, also $\text{Var}(p_1) = p(1-p)/n \doteq 0.126/n$

- generell hilfreich: Symmetrie ausnutzen
  - falls $P(X \leq -x) = P(X \geq x)$, dann sollte man auch die negativen Realisationen nutzen
  - im Beispiel: p2←**mean**((**abs**(X)>2))/2
  - hier: $Y_i^s = \mathrm{I}\{|X_i| > 2\}/2 \sim \mathrm{Bin}(1, 2p)/2$, also $\mathrm{Var}(p_2) = p(1-2p)/(2n) \doteq 0.052/n$
  - im Beispiel: weiß, dass $1 - 2p = 2\int_0^2 g(x)\,\lambda(dx)$, $\rightsquigarrow$ Z←**runif**(n,**min**=0,**max**=2); p3←1/2−2∗**mean**(g(Z))
  - hier: $\mathrm{Var}[f(Z)] = \frac{1}{2}\int_0^2 g^2\,d\lambda - (\frac{1}{2}\int_0^2 g\,d\lambda)^2$, also $\mathrm{Var}(p_3) \doteq 0.0284/n$.
- oft hilfreich: Variablentransformation
  - im Beispiel: mit der Variablentransformation $y = 1/x$ erhalten wir zufällig $p = \int_0^{1/2} g\,d\lambda$, $\rightsquigarrow$ Z1←**runif**(n,**min**=0,**max**=1/2); p4←**mean**(f(Z1))/2
  - hier: $\mathrm{Var}[f(Z')] = 2\int_0^{1/2} g^2\,d\lambda - (2\int_0^{1/2} g\,d\lambda)^2$, also $\mathrm{Var}(p_4) \doteq 3.8\mathrm{E}{-4}$.

- Importance Sampling
  - Idee: Masse auf "interessanten" Bereich verschieben
  - Fakt:
    Seien P und Q Maße auf $\mathbb{B}$ mit Dichten p, q, so dass
    $Q(A) = 0 \;\Rightarrow\; P(A) = 0$.
    Gesucht $I = \int \phi(x)p(x)\,\lambda(dx)$; aber $I = \int \psi\, dQ$ mit
    $\psi = \phi p/q$;
    daher ist I schätzbar durch $I_Q := \mathrm{mean}(\psi(Y_i))$, $Y_i \overset{\text{u.i.v.}}{\sim} Q$
    und $\mathrm{Var}[I_Q]$ wird minimal genau dann, wenn $q \propto \phi p$.
  - im Beispiel $\mathrm{I}_{\{x>2\}}\, g(x) \approx 2\,\mathrm{I}_{\{x>2\}}/x^2$;
    dies ist Dichte der Variable Z2←2/**runif**(n);
- Antithetische Variablen
  - Idee: Ausnützen negativer Korrelationen
  - Fakten
    * $\mathrm{Var}[X+Y] =$
      $\mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Corr}[X,Y]\sqrt{\mathrm{Var}[X]\,\mathrm{Var}[Y]}$;

$\Longrightarrow$ negatives $\mathrm{Corr}[X, Y]$ reduziert die Varianz!

* $U \sim \mathrm{ufo}(0, 1) \Rightarrow U' = 1 - U \sim \mathrm{ufo}(0, 1),\ \mathrm{Corr}(U, U') = -1/4$
* Damit sind dann aber auch die nach F verteilten Variablen $Y = F^{-1}(U)$, $Y' = F^{-1}(U')$ negativ korreliert

- Kontroll/Regressions–Variablen
  - oft können wir uns "billig" Kovariate $W_1, \ldots W_p$ verschaffen, die selbst viel der Variabilität der zu integrierenden Variable Z erklären
  - Genauer ist ein lineares Regressionsmodell in den Variablen $\beta_0, \ldots \beta_p$ einzupassen, nämlich mit $\hat{I} = -\hat{\beta}_0$

$$Z = \beta_0 + \sum_{i=1}^{p} \beta_i (W_i - \mathrm{E}[W_i]) + \varepsilon$$

  - im Beispiel $g(x) \approx \alpha_1 x^2 + \alpha_2 x^4$, also $W_1 = Z^2$, $W_2 = Z^4$, $I = 1/2 - \int_0^2 g \, d\lambda$

$$f(Z) = \frac{1}{2} + \beta_0 + \beta_1 (Z^2 - 8/3) - \beta_2 (Z^4 - 32/5)$$

  - hier $n \, \mathrm{Var}[\beta_0] \approx 6.3\mathrm{E}-4$
  - entsprechendes Modell für $I = \int_0^{1/2} g \, d\lambda$ ergibt $n \, \mathrm{Var}[\tilde{\beta}_0] \approx 1.1\mathrm{E}-9$.

2.8.4 globale Optimierung

- gesucht: globales Min[Max]imum einer Funktion $f : \mathbb{R}^k \to \mathbb{R}$
- Problem: unter mehreren lokalen Extrema das globale zu finden
- weitere mögliche Komplikationen:
  - Funktion nicht glatt genug für differenzielle Methoden,
  - Auswertung der Ableitung ist zu "teuer"
- Idee: "gleichberechtigte" Auswahl aller potenzieller Auswertungsstellen $\rightsquigarrow$ stochastische Auswahl
- auch Kombination aus globaler (stochastischer) Methode und lokaler (numerischer) Methode möglich
- Dilemma: alle Bereiche abdecken $\rightarrow\leftarrow$ Konzentration um die potenziellen Extrema
- raffinierte Techniken — hier nur als Schlagworte
  - Simulated Annealing
  - Sintflut Algorithmus
  - MCMC-Methoden / Gibbs–Sampling
  - genetische Algorithmen

2.9 Resampling–Techniken

2.9.1 Idee

- in realen Datenbeständen oft "zu wenig" Daten
- künstlich simulierte Daten würden ein Modell unterstellen, das aber ist nicht bekannt
- Kann man die Information, dass es sich um $\mathrm{u.i.v.}$ Daten handelt, nutzen, um mehr an Informationen aus der Stichprobe zu holen?
- JA: die Daten hätten ja genauso auch in einer anderen Reihenfolge eintreten können
- $\rightsquigarrow$ Resampling–Techniken
- erst relativ kurz bekannt, da i.a. sehr rechenaufwendig

2.9.2 Jack-Knife

- geht zurück auf Quénouille und Tukey
- Ziel: Biasreduktion
- Definition:
  - Seien $X_1, \ldots, X_n \overset{\text{u.i.v.}}{\sim} F$
  - Sei T_n ein *linearer Schätzer* in $X_1, \ldots, X_n$ in dem Sinn, dass $T_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$ mit einer (nicht notwendig linearen) Funktion $f : \mathbb{R} \to \mathbb{R}$
  - Sei $T_{n;-j} := \frac{1}{n}\sum_{j\neq i=1}^{n} f(X_i)$
  - Sei $T^*_{n;j} := nT_n - (n-1)T_{n;-j}$ ein *Pseudo–Summand*.
  - Dann: $T_n^{(J)} := \frac{1}{n}\sum_{i=1}^{n} T^*_{n;i}$ heißt *Jackknife*–Schätzer zu T_n

- Eigenschaften:
  - Varianz:

$$S_n^{(J)\,2} := \mathrm{Var}[T_n^{(J)}] = \frac{1}{n}\sum_{i=1}^{n}\frac{(T_{n;i}^{*} - T_n^{(J)})^2}{n-1}$$

  - Biasreduktion: Sei $\mathrm{Bias}[T_n] := \frac{a}{n}$. Dann ist $T_n^{(J)}$ biasfrei!
- Konfidenzintervalle:
  - Heuristik (— nicht immer korrekt) $\mathcal{L}(\frac{T_n^{(J)}-\theta}{S_n^{(J)}}) \approx t_{n-1}$
  - daher $100(1-\alpha)\%$–Konfidenzintervall gegeben durch $T_n^{(J)} \pm S_n^{(J)}\,(t_{n-1})^{-1}(1-\alpha/2)$

2.9.3 Bootstrap

- Literatur: Efron and Tibshirani (1993) — inklusive R–code: [Windows] [Linux]
- Name:
  Münchhausen — am eigenen Schopf aus dem Sumpf ziehen

- Idee:
  alle Wertekombinationen der Realisationen $x_1^\natural, \ldots, x_n^\natural$ der Original–Stichprobe $X_1, \ldots, X_n$ — mit Mehrfach–Replikationen — sind genauso wahrscheinlich wie das Original
- ergibt n^n Möglichkeiten — viel zu viele schon für kleine n
- Umsetzung:
  Ziehe aus den Realisationen der Original–Stichprobe $X_1, \ldots, X_n$ **mit Zurücklegen** $j = 1, \ldots, N$ Stichproben $X_1^{(j)}, \ldots, X_m^{(j)}$ der Länge m (nicht notwendig n)
- ein $100(1-\alpha)\%$–Konfidenzintervall:
  $m := n$, T_j Auswertung des Schätzers an der Bootstrap–Stichprobe j.
  Dann ist $[T_{[\alpha/2\, N:N]}; T_{[(1-\alpha/2)N:N]}]$ mit $T_{[i:n]}$ der i-t-größten Realisation unter den T_j, $j = 1, \ldots, N$ ein $100(1-\alpha)\%$ Konfidenzintervall.

R–Beispiel 2.9-1 [Konfidenzintervall für den Median]:

- Situation: Wir haben eine Stichprobe $X_1, \ldots, X_n \overset{\text{u.i.v.}}{\sim} F$ mit einem unbekanntem F.
- Ziel: Wir wollen den Median m von F schätzen.
- Methode: Schätzung durch den Stichprobenmedian $\hat{m}$ — **median**(X)
- Problem: Wie genau ist die Schätzung?
- asymptotische Theorie: Mit $f(m)$ der Dichte von F ausgewertet im Median von F gilt für $n \to \infty$

$$\sqrt{n}(\hat{m} - m) \circ F^n \xrightarrow{\text{w}} \mathcal{N}(0, \frac{1}{4f(m)^2})$$

- aber: F, bzw. f nicht bekannt.
- Beispiel–Datensatz: Eruptionen des Old Faithful, mit Median 4

Lösung 1: Dichteschätzung

```
library(MASS)
data(faithful)
attach(faithful)
density(eruptions,n=1,from=4,to=4.01,
        width=0.41)$y
density(eruptions,n=1,from=4,to=4.01,
        width=0.63)$y
#Schaetzung der Streuung des Medians
#durch Dichte--Schaetzung
#f(m) ~ 0.415
1/(2*sqrt(length(eruptions))*0.415)
```

Lösung 2: Bootstrap

```
#Schaetzung der Streuung des Medians
#durch 1000 Bootstrap--Stichproben
```

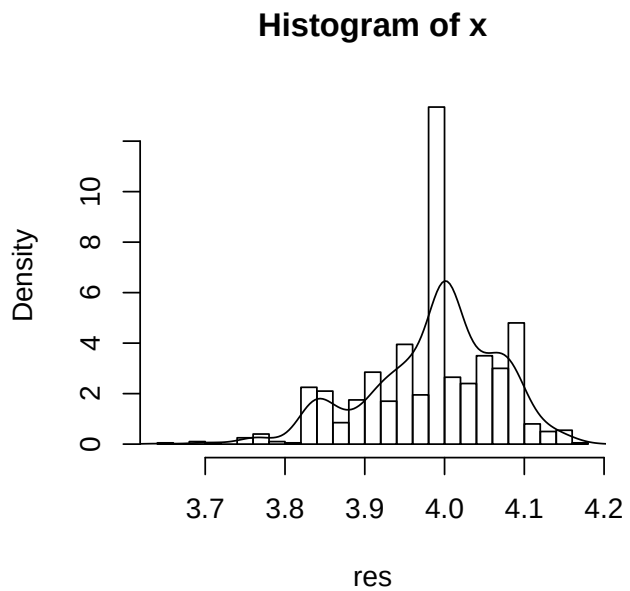

```
#
set.seed(101);m<-1000
# zur Reproduzierbarkeit
res<-numeric(m)
for(i in 1:m)
     res[i]<-median(
          sample(eruptions,replace=T))
#Bias:
mean(res-median(eruptions))
#Varianz:
sqrt(var(res))
```

Die Bootstrap–Verteilung ist nicht normal:

```
hist.FD(res,prob=T)
lines(density(res,n=200,width=
       bandwidth.nrd(res)))
```


```
#Bandweitenwahl
c(ucv(res),bcv(res))
width.SJ(res)
# 95% Bootstrap-Konfidenzintervall
quantile(res,c(0.025,0.975))
```

Histogramm zum ge–bootstrap–ten Median der Eruptionen des Geysirs Faithful



Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

2.9.4 Bagging und Boosting

- verfeinerte Versionen des Bootstrap (Methoden zur Klassifikation und Regression)
- Situation: Daten $(X_1, Y_1), ..., (X_n, Y_n)$, wobei $X_i \in \mathbb{R}^p$ (p groß!!) und $Y_i \in \mathbb{R}$
- allgemeine Vorgehensweise: Gewichtung der Daten, Ziehen von Bootstrap-Stichproben (i.d.R. Ziehen mit Zurücklegen), Bestimmung der Schätzer, Aggregation der Schätzer, d.h. Berechnung eines gewichteten Mittels über die Schätzer
- Varianten: Bagging (**B**ootstrap **agg**regat**ing**), Subagging, Boosting, AdaBoost, L_2-Boosting, Bag-Boosting, ...
- Bagging und Boosting Verfahren reduzieren Bias und Varianz, d.h. erhöhen die Schätzgenauigkeit
- Literatur und genauere Informationen unter: www.boosting.org

3 Programmierung

Quellen: Neben Venables and Ripley (1999), Chapter 4 auch

R Language Definition

3.1 Kontrollstrukturen

3.1.1 Gruppierung von Befehlen: Blöcke

- verschiedene Anweisungen in einer Zeile können durch ";" gruppiert werden
- eine Folge von Anweisungen wird durch Klammerung mit"`{<block>}`" zu einem (Anweisungs–)Block

3.1.2 Bedingte Ausführung von Blöcken

3.1.2 (a) if – Befehl

- Syntax

```
if (<Bedingung>)
        <Anweisung(sblock)1>
else
        <Anweisung(sblock)2>
```

- der Teil ab "**else**" kann weggelassen werden
- Achtung: Man beachte die Klammerung der Blöcke bei Einlesen des Codes mit **source**

```
if (0) { print(1); }
else { print(2); } ### syntax error

if (0) { print(1); } else { print(2); } ## oK
```


```
if (0) { print(1);
} else { print(2); } ## oK
```

- in der Bedingung kann ein beliebiger (skalarwertiger) logischer Ausdruck stehen (siehe Abschnitt 1.4.5)
- R–Beispiel 3.1-1 [Test auf Symmetrie einer Matrix X]:

```
#exakt
if (all(X == t(X)))
    cat("Matrix symmetrisch")

#nahezu bis auf eps
if (all.equal.numeric(X, t(X)))
    cat("Matrix nahezu symmetrisch")
```

- beachte Verwendung von **all.equal**, **all.equal.numeric**, um bei vielen Vergleichen nur **einen** logischen Wert zurückzubekommen

3.1.2 (b) Unterscheide: ifelse – Befehl

- Syntax **ifelse** (<Bedingung>,true.value,false.value)
- auch vektorwertig möglich!
- R–Beispiel 3.1-2 $[x \log(|x|)]$:

```
#mit Warnung
x.logx<- ifelse (x==0,0,x*log(abs(x)))
#ohne
x.logx<-x * log(abs(x)+(x==0))
```

- in solchen Situationen oft noch besser: Indikatorfunktionen, c.f. Abschnitt 3.2.3
- aber Achtung: bei vektorwertigen Zuweisungen weist **ifelse** nur die Koordinaten zu, die zur Dimension der Bedingung passen

```
x <- 3
```


```
(y ← ifelse(x<3,1:3,3:5))
(y ← if(x<3) 1:3 else 3:5)
```

3.1.2 (c) switch – Befehl

- zur Vermeidung vieler paralleler `if`–Alternativen
- Syntax **switch** `(<test>,<Alternativenliste>)`
- Ist der Wert `<value>` von `<test>` eine ganze Zahl zwischen 1 und **length**`(<Alternativenliste>)`, wird Anweisung(sblock) Nummer `<value>` ausgeführt, sonst Rückgabe von NULL
- Ist der Wert `<value>` von `<test>` ein String, so wird das Listenelement mit Namen `<value>` ausgeführt, sonst Rückgabe von NULL
- Verwendung einer Default–Anweisung durch Einschluss einer Alternative ohne Namen am Ende der Liste
- Abkürzungen der Namen mit **pmatch**

- R–Beispiel 3.1-3 [verschiedene Tests]:

```
# in test stehe der Name des
#    auszuwertenden Tests,
# in y die Daten
#
# (genauer geht es um Tests auf
#  Varianzgleichheit; als
#  Alternativen stehen
#  Levene, Cochran und Bartlett
#  zur Auswahl)
#
#################################

# brutal
result<- if (test=="Levene") levene(y,f)
         else
           if (test=="Cochran") cochran(y,f)
```


```
            else bartlett(y,f)
# switch1
#
result← switch(test, "Levene"=levene(y,f)
         "Cochran"=cochran(y,f),
         bartlett(y,f))
# switch2
# mit mehreren Schreibweisen fuer eine
# Alternative
result← switch(test, Levene=, levene=,
         "Levene's␣test"=levene(y,f),
         Cochran=, cochran=,
         "Cochran's␣test"=cochran(y,f),
         Bartlett=,bartlett=,
         "Bartlett's␣test"=, bartlett(y,f))

# switch3:
```


```
# mit Abkuerzungen
result <- switch(pmatch(test,c("Levene",
          "levene","Cochran","cochran",
          "Bartlett=","bartlett",nomatch=""),
          "1"=,"2"=levene(y,f),
          "3"=,"4"=cochran(y,f),
          bartlett(y,f))
```

3.1.3 Schleifen

3.1.3 (a) for – Befehl

- Syntax **for** (<Variable>in <Sequenz>) <Anweisung(sblock)>
- die Schleifenvariable (<Variable> durchläuft mit den Schleifendurchgängen die Zahlenfolge <Sequenz> — meist dargestellt als <von>:<bis> oder **seq**(along=X)

3.1.3 (b) Sprünge: break und next

- mit **break** kann jederzeit die aktuelle Schleife verlassen werden
- mit **next** springt man an den Beginn des nächsten Schleifendurchlaufs

3.1.3 (c) while – Befehl

- Syntax **while** `(<Bedingung>)` `<Anweisung(sblock)>`
- in der Bedingung kann ein beliebiger (skalarwertiger) logischer Ausdruck stehen (siehe Abschnitt 1.4.5)

3.1.3 (d) repeat – Befehl

- Syntax **repeat** `<Anweisung(sblock)>`
- Ausstieg nur mit "**break**"

R–Beispiel 3.1-4 [Schleifen]:

```
# Ausdruck 1:10
### FOR
for (i in 1:10) print(paste("Zahl Nr. ",i))
### WHILE
i←0
while(i<10) {i←i+1;print(paste("Zahl Nr. ",i))}
### REPEAT
i←0
repeat {i←i+1
        if (i>10) break
            else print(paste("Zahl Nr. ",i))
        }
```


3.2 Vermeidung von `for`-Schleifen

3.2.1 Schleifen sind langsam!

3.2.1 (a) `S` — eine Interpreter–Sprache

Im Gegensatz zu Programmiersprachen wie `FORTRAN`, `PASCAL`, `C` und `C++` wird ein Programm vor dem Start **nicht** compiliert, sondern zur Laufzeit *interpretiert*.

Das hat zur Folge, dass der Anweisungsblock in einer Schleife immer wieder neu übersetzt wird, was sich stark in der Laufzeit bemerkbar macht.

3.2.1 (b) Paradigma: matrixorientierte Programmierung

Verwendet man aber die vektorwertigen Funktionen aus R, so greifen diese auf Indexoperationen zurück, die "ganz unten", maschinennah implementiert sind. Als Nachteil ergibt sich unter Umständen ein enormer Speicherbedarf.

3.2.2 ein Beispiel: Blatt 5 Aufgabe 1

Link aufs File

```
##################################################
#  Blatt 5 Aufgabe 1 vektorwertig
##################################################


# Formulierung mit for
#
luecke.for<-function(n)
{X.x<-runif(n)
 X.y<-runif(n)
 d<-2
 # zum Vergleich treten nur die Zeilen
 # 1 : n-1 an
```


Peter Ruckdeschel
Matthias Kohl

```
  for(i in 1:(n-1))
   { # zum Vergleich treten nur Eintraege
     # mit Index >i an
    for (j in (i+1):n)
      {d0<-(X.x[i]-X.x[j])^2+
           (X.y[i]-X.y[j])^2
       d<-c(d,sqrt(d0))
      }
    }
  return(min(d))
  }


# vektorwertige Formulierung
#
luecke.vec<-function(n)
{#erst die x Koordinate
```


```
X.x<-runif(n)

#erzeuge eine Matrix mit identischen Spalten
X.xx1<-X.x%o%(0*X.x+1)
# Matrtix mit identischen Zeilen
X.xx2<-t(X.xx1)

# Matrix mit Eintraegen (X_i-X_j)^2_{i,j}
X.xd<-(X.xx1-X.xx2)^2

# Diagonale aus Minimumbildung ausschliessen
X.xd[row(X.xd)==col(X.xd)]<-2

#Platz machen
rm(X.xx1,X.xx2)

#das gleiche fuer die y Koordinate
```


```
  X.y<-runif(n)
  X.yy1<-X.y%o%(0*X.y+1)
  X.yy2<-t(X.yy1)
  X.yd<-(X.yy1-X.yy2)^2
  X.yd[row(X.yd)==col(X.yd)]<-2
  rm(X.yy1,X.yy2) #Platz machen

  #Addition der Koordinaten
  X.d<-X.xd+X.yd
  return(min(sqrt(X.d)))
  }

#Rahmenfunktion
#
#
blatt3.4<-function(M=500,for.vec=1,
                   n=c(5,25,100,500),out=0)
```


```
{
# Achtung am geschicktesten fuer
# Speicherverwaltung, siehe Kapitel 9
# statisches Deklarieren der Matrix,
# in die die Ergebnisse kommen
#
d<-matrix(0,M,length(n))

# fuer jedes n fuehre die
# M Wiederholungen durch
for(j in 1:length(n))
   {
   k<-n[j]
    # M Wiederholungen
   for(l in 1:M)
       {if ((out==1)&&(l%%10==0)) cat(l,"\n")
        if (for.vec==1)
```


```
            d[l,j]<-luecke.vec(k)
        else
            d[l,j]<-luecke.for(k)
        }
    if (out==1) print(d)
    # Zusammenkleben der Ergebnisse
    }
  colnames(d)<-paste(rep(paste("n =",k),length(n)))
  summary(as.data.frame(d))
  boxplot(as.data.frame(d))
  return(d)
  }

# Zeitvergleich
print(system.time(blatt3.4(M=40,n=50,for.vec=1)))
print(system.time(blatt3.4(M=40,n=50,for.vec=0)))
```


```
####################
 # fuer Blatt 3
print(system.time(d<-blatt3.4(out=1)))

summary(d)

boxplot(as.data.frame(d))

#Mediane
apply(d,2,median)
#IQR's
apply(d,2,IQR)

par(new=T)
plot(d[,1])
par(mfrow=c(2,2))
```


```
#Histogramme
apply(d,2,hist)

#Dichteplot
plde<-function(x,...){plot(density(x,...))}
apply(d,2,plde)
```


3.2.3 Tricks zur Vermeidung von Schleifen

Peter Ruckdeschel
Matthias Kohl

3.2.3 (a) Verwedung von Indikatorfunktionen

- anstelle von **if**– Fallunterscheidungen nach Möglichkeit **ifelse** – Anweisungen verwenden
- noch besser: Arbeit mit logischen Indikatorfunktionen oder **pmin**, **pmax** z.B. für die Funktion

$$x \mapsto H(x) := \begin{cases} -b & x < -b \\ x & \text{für } -b \leq x < b \\ b & b \leq x \end{cases}$$

```
norm1 <- pmax(b/2, norm(x))
H <- x*pmin(1, b/norm1)
# kein Teilen durch 0!
```

oder

```
w←x*0+1
w[norm(x)<b]←b/norm(x)
H←x*w
```

3.2.3 (b) Matrix–Multiplikation

siehe Beispiel 3.2.2

3.2.3 (c) outer

siehe Abschnitt 1.7.2 und Beispiel 1.7-2

3.2.3 (d) geschichtete Stichproben

- Situation:
  - will Genauigkeit einer Stichprobe von Umfang `Nmax * N`
  - maximale auf einmal matrixwertig zu behandelnde Dimension des Problems: `Nmax`

- Lösung: *stratifizierte* (=geschichtete) Stichprobe
  - **for**–Schleife der Länge `N`
  - jeder Schleifendurchlauf produziert ein Resultat der Genauigkeit einer Stichprobe von Umfang `Nmax`
  - aus den `N` Teilergebnissen durch Mittelung (**mean**) Genauigkeit einer Stichprobe von Umfang `Nmax * N`

3.2.3 (e) FFT

- in einigen Kontexten keine unmittelbare vektorwertige Formulierung möglich, z.B.
  - bei Rekursionen im Zeitreihenkontext
  - bei Faltungen zur Berechnung der Verteilung von Summen
- in beiden Beispielen möglich: Übergang zur Fouriertrafo
  - denn: Fouriertrafo der Faltung ist Produkt der Fouriertrafos der Faltungsfaktoren
  - dort dann vektorwertiges Kalkül möglich

- noch besser: FFT (*Fast Fourier Transformation*) — Tukey
  - benutzt trigonometrische Rekursionen
  - nur Berechnung der Koeffizienten mit Index der Form 2^k nötig

3.3 Schreiben von Funktionen

3.3.1 Syntax

3.3.1 (a) Bestandteile einer Funktion

- *body*: der S-Code
- *formal arguments* die Argumente
- *environment*: ein Paar aus einem *frame* und eine *enclosure*
  - *frame*: eine Liste von Paaren aus Symbol-Name und Wert — die "lokalen" Variablen und deren Werte
  - *enclosure*: ein Zeiger auf ein umschließendes *environment*

– für die letzten beiden Begriffe siehe auch Abschnitt 3.6

3.3.1 (b) Rückgabewert

- durch **return**`<rueckgabewert>` oder der letzte ausgewertete Ausdruck
- Rückgabe von mehreren Argumenten durch Listenbildung, am besten vom Typ
  return(**list** (`<argname1>=<arg1>,<argname2>=<arg2>,....`))
- bei größeren Rückgabeobjekten: besser noch in der Funktion einer Variablen zuweisen $\rightsquigarrow$ "schlankere" Übergabeobjekte

3.3.1 (c) Argumente

- Zweck: Übergabe von Parametern aus der aufrufenden Umgebung
- Setzen von Defaultwerten durch
  `<arg.name>=<arg.default.wert>`
  c.f. Abschnitt 1.2.4; dann möglich: *lazy calling*

- Fehlende Argumente:
  - Argument `X` wird nicht mit übergeben
  - Abfangen im Funktionskörper durch **missing**–Abfrage
  - Beispiel:

```
if (missing(X))  Y←0
   else  Y←min(X)
```

- spezielles Argument " ... "
  - steht für beliebig viele weiter mit übergebbare Parameter
  - diese können dann an andere Funktionen, die im Funktionenkörper aufgerufen werden, übergeben werden
  - nützlich beim Übergeben von Funktionen als Parameter
  - siehe auch dichte.r
- Übergabe von Funktionen

Um Verwechslungen mit dem Auslassungssymbol auszuschließen, verwenden wir für dieses in Zukunft immer "...."

– jederzeit möglich durch Übergabe des Funktionsnamens,
– siehe auch dichte.r

- Matching der Argumente — Reihenfolge
  - erst: *exaktes Matching*:
    alle Argumente werden belegt, die mit einem exakt passenden Namen versehen werden;
    daher notwendig Eindeutigkeit von
    * Argument–Namen in der Deklaration
    * Argument–Namen im Aufruf
  - dann: restliche Argumente mit *partiellem matching*
    * matching sobald der deklarierte Argument–Namen–Anfang eindeutig mit übergebenem Namen übereinstimmt
    * Bsp: `f←function(fumble,fooey)`
      · falsch: `f(f=1,foo=2)`
      · korrekt: `f(f=1,fooey=2)`
    * bei Argument `...` : partielles matching mit allen Argumenten davor (in der Deklaration)
  - schließlich *positionelles Matching*:
    * der Rest der unbenannten übergebenen Argumente wird in

der Deklarationsreihenfolge den verbleibenden Argumenten zugewiesen

  - bei Argument ... : Sind noch übergebene Argumente übrig, so werden diese im Funktionenkörper in der Anweisungsreihenfolge noch freien Übergabeparametern aufgerufener Funktionen zugewiesen.

- Argumentauswertung: *lazy evaluation*
  - bei Aufruf einer Funktion wird ein neuer *evaluation frame* erzeugt, vgl. Abschnitt 3.2
  - übergebene und default–Argumente werden unterschiedlich behandelt:
    - übergebene Argumente werden in der aufrufenden Umgebung ausgewertet
    - default–Argumente werden in der lokalen Funktions–Umgebung ausgewertet
    - Übergabe von Argumenten per *call–by–value*, es wird also eine lokale Kopie angelegt, die (bei Bedarf) mit dem Wert

des übergebenen Arguments aus der aufrufenden Umgebung initialisiert wird

- *lazy evaluation*
  - bei Aufruf der Funktion werden die Parameter noch nicht initialisiert ($\rightsquigarrow$ "lazy")
  - daher passiert die Wertzuweisung `foo(x=y)` erst, wenn die Variable `x` im Funktionenkörper gebraucht wird
  - bei Aufruf wird die bei Bedarf auszuführende Initialisierung in ausführbarem S–code (Text) in einer sogenannten *promise* abgelegt.
  - Zugriff auf die *promise* mit **substitute** ergibt einen Rückgabewert vom Typ `expression`

* Umwandlung des von **substitute** erhaltenen Codes durch **deparse**
*

R–Beispiel 3.3-1 [Plot mit automatischem Titel]:

```
myplot ← function(x,y){
    lab ← deparse(substitute(y))
      ....
    title(main=paste("Ein␣Plot␣von␣",lab))
      ....
    }
```

– globale vs. lokale Variablen
  * alle Variablen innerhalb des Funktionenkörpers sind lokal;
  * alle Variablen aus der aufrufenden Umgebung stehen bei Bedarf als lokale Kopien zur Verfügung, sofern sie nicht durch lokale überladen worden sind;
  * Zuweisungen im Funktionsaufruf, wie foo(x←y) verändern

die Variable `x` der aufrufenden Umgebung

* Zugriff / und Manipulation von globalen Variablen innerhalb eines Funktionenkörpers durch `assign("<objname>",<Wert>,frame=0)`, c.f. Abschnitt 3.6

3.3.2 Editieren von Funktionen

- am besten speichert man Funktionen als R–Skripten, möglicherweise zusammen mit weiterem R–Code
- es gilt das in Abschnitt 1.10.2 gesagte
- weitere Möglichkeiten zur Manipulation von R–Objekten:
  - **fix** `(<obj.name>)`: dabei wird der Editor aus **options**`(editor=«editorname>")` verwendet; nach Speichern liegt das manipulierte Objekt im Speicher
  - `<obj.name>←ed(<obj.name>,editor=«editorname>")` leistet dasselbe

3.3.3 Fehlerbehandlung

3.3.3 (a) warning

- gibt eine Warnung bei Rückkehr zum `session`–Modus zurück
- unterbricht das Programm nicht

3.3.3 (b) stop

- gibt eine Fehlermeldung bei Rückkehr zum `session`–Modus zurück
- unterbricht das Programm
- bricht aber die `session` nicht ab
- erzwingt Abbruch, falls notwendiger Parameter fehlt, z.B.
  rpois←**function**(n,**lambda**=**stop**("no␣lambda␣arg")){....}

3.3.3 (c) missing

- vgl. Abschnitt 3.3.1 (c)

3.3.3 (d) on.exit

- wird erst beim Verlassen der Umgebung ausgeführt (egal ob nach korrekter Beendigung oder per Error)
- nützlich zum lokalen Setzen von Optionen, z.B.

```
oldpar ← par(...)
on.exit(par(oldpar))
```

- Löschen der momentanen **on.exit** Anweisung durch **on.exit()**

3.3.3 (e) try

- Situation: möchte innerhalb einer Routine eine andere (fremde) aufrufen, die je nach Situation mit Fehler abbricht oder den richtigen Wert ergibt

- Problem: mit `stop` oder `warning` entweder ganz heraus oder gar nicht; kein individuelles Abbruchhandling möglich, da kein Zugriff auf innere Routine
- Ablauf in "Sandkasten" $\rightsquigarrow$ `try`
- Achtung:
  - funktioniert nur innerhalb von Routinen, nicht im Kommando–Prompt
  - man sollte wissen, welche(r) Fehler passieren und warum
- Vorteil: verhindert Abbruch des Programms, man kann steuernd eingreifen
- Syntax: **try**(<expr>, first =TRUE)
- Rückgabewert: entweder Ergebnis von Expression oder Fehler–Meldung, die weiterzuverarbeiten ist
- durch Setzen von **options**(**show**.error.messages=FALSE) vor Aufruf von `try` wird Ausgabe von Fehlermeldung unterdrückt

- Achtung: nach Aufruf von `try` wieder auf `TRUE` setzen!
- R–Beispiel 3.3-2 [Beispiel try]:

```
try.test ← function(x)
{
 options(show.error.messages = FALSE)
 erg ← try(sqrt(x))
 options(show.error.messages = TRUE)

 if(is.numeric(erg)==F)
 {
  cat("keine␣gueltige␣Zahl␣eingegeben!\n")
  return(NA)
 }

 return(erg)
}
```


```
erg <- try.test(10)
print(erg)

erg <- try.test("a")
print(erg)
```

3.3.4 Hilfe-Files / Dokumentation

- Soll der eigene Code auch von anderen genutzt werden, ist es sinnvoll, eine Online–Hilfe zu schreiben — vgl. auch Referat
- erster Schritt: mit Befehl **prompt**(<objname>)
  - erzeugt eine Help–Schablone als File "<objname>.Rd" im aktuellen Verzeichnis
  - siehe auch *"Writing R documentation"* in "Writing R Extensions" und *"Guidelines for Rd files"*, sowie Referat

- zweiter Schritt: Editieren der Schablone in einem Editor
- dritter Schritt: Kopieren in "`$R_HOME/src/library/base/man/`" — ohne Endung `.Rd`

3.4 Debugging

- zwei unterschiedliche Ausgangssituationen
  - post mortem: das Programm ist bereits abgestürzt (Inspektion ausgelöst durch Signal "Error")
  - Inspektion ausgelöst durch in Code eingefügtes Signal
- weitere Quellen: Writing R Extensions (2006b), Kap. 4, R Language Definition (2006c), Kap. 9, Bavington (2003), Peng (2002):

3.4.1 Exkurs: Exception-Handling in R

- Fehler/Warnings sind Spezialfälle für eine *Exception* — ein Situation außer der Reihe
- *Exception-Handling* funktioniert nicht, falls der R/S-Plus -Prozess wird abgeschossen oder gar das System abstürzt
- sonst bei "regulärer" Exception: Interpreter erhält ein Signal, wieder ein S-Objekt!, genauer der Klasse `condition`
- in Abhängigkeit des Signals wird Ausführung (*execution*) weiteren "Konsequenz"-Codes ausgelöst (eigentliches Exception-Handling)
- in diesen "Konsequenz"-Code kann man eigenen S-Code einhängen, vgl. ?withRestarts
- ein Signal der condition *warning* löst per default keinen Abbruch aus, kann aber mit **options**(warn=2) zu Signal der condition *error* gemacht werden, vgl. Tabelle 1.10-2

3.4.2 Post-mortem Analyse

- bei Signal der condition *error* wird anschließend Code abgearbeitet, der in der Option `error` festgelegt ist; per default: NULL
- in jedem Fall gibt **traceback** den Stapel (stack) der letzten Aufrufebenen wieder

R–Beispiel 3.4-1 [ein einfacher Fehler]:

Situation:

- **log2**(-1)= $\log_2(-1)$ erzeugt NaN und eine Warnung;
- aber hier: bei Warnung wird Abbruch erzwungen

Ergebnis

```
> options(warn=2)
> log2(-1)
Error in log(x, base) : (converted from warning) NaNs produced

> traceback()
6: doWithOneRestart(return(expr), restart)
5: withOneRestart(expr, restarts[[1]])
```


```
4: withRestarts({
       .Internal(.signalCondition(simpleWarning(msg, call), msg,
           call))
       .Internal(.dfltWarn(msg, call))
   }, muffleWarning = function() NULL)
3: .signalSimpleWarning("NaNs produced", quote(log(x, base)))
2: log(x, 2)
1: log2(-1)
```

- Alternative zu NULL in **options**("error"=NULL): **dump**.frames — und nur in S-Plus: **dump**.calls; Beispiel:;
  - ähnlich wie **dump** legen **dump**.calls und **dump**.frames ein Abbild von Teilen des Arbeitsspeichers an — auch auf File, falls gewünscht
  - **dump**.calls speichert den Keller (Stack) der ineinander geschachtelten Funktionsaufrufe (siehe 1:–6: in Beispiel 3.4-1)
  - **dump**.frames legt zusätlich noch die entsprechenden Frames ab (siehe dazu auch Abschnitt 3.6)
  - ein dump von **dump**.frames ist dann auswertbar dann mit **debugger**

- **debugger**
  - umfassendes System zur schrittweisen Rückverfolgung eines Fehlers
  - operiert auf zwei Ebenen
    A der Liste der aktuell im Speicher liegenden *frames* — entschieden durch `Selection:`
    B innerhalb eines aktuell im Speicher liegenden *frames*: kenntlich gemacht durch Prompt `Browse[n]`, n die Ebene des Calls, für die man sich in `Selection:` entschieden hat
  - offline, indem ein bereits ge"`dump`"–ter `last.dump` analysiert wird
  - interaktiv durch Abfragen von Optionen
  - Ebene A:
    * es werden alle Alternativen durchnummeriert dargestellt
    * im anschließenden `Selection`–tag gibt man die entsprechende Nummer der gewünschten Alternative an
    * die Nummer `0` springt eine (*frame*–)Ebene nach oben, bis

schließlich auf *session*–Ebene

  - Ebene B — Debugger-Modus:
    * beliebiger S-Code kann eingegeben werden
    * insbesondere: Werte von Variablen (innerhalb dieses *frames*) können inspiziert werden
    * spezielle Befehle, die R speziell interpretiert — siehe Tabelle 3.4-2

- Alternative zu **dump**.frames und **debugger**:
  options("error"=recover)
  - im wesentlichen das gleiche — nur ohne Umweg mit einem dump (also ohne Variablen zu kopieren)
  - in diesem Sinn nur interaktiv sinnvoll

Tabelle 3.4-2 [Sonderbefehle im Debugger-Modus]:

wichtig für **browser**, **debug**, recover, **debugger**

⟨RET⟩	beim Debuggen gehe zum nächsten Ausdruck bzw. beim Browsen führe den ursprünglichen Code weiter aus
`c`	"cont" — führe den ursprünglichen Code weiter aus
`n`	führe den nächsten Ausdruck aus
`where`	zeige die Hierarchie der aufgerufenen Funktionen (call stack)
`Q`	breche ab und springe zum Top-Level

Zugriff auf "klassische" Objekte `c`, `n`, `Q` mit **get**, bzw. mit **print**

3.4.3 Selbst ausgelöste Exceptions

3.4.3 (a) browser

- ist in eigenen Code als Befehl zu integrieren
- unterbricht den Ablauf des Programms und springt in Debugger-Modus
- Ermöglicht Inspektion / Manipulation aller Objekte der aktuellen Umgebung
- Rückkehr zum Ablauf mit `0`

3.4.3 (b) debug

- Syntax: **debug**`(<fun>)` markiert eine Funktion `<fun>` zum Debuggen
- bei jedem Aufruf von Funktion `<fun>` Sprung in Debugger-Modus
- Aufheben der Markierung durch **undebug**

3.4.3 (c) trace

- Syntax: **debug**(what,tracer,exit,at,print,....) markiert eine Funktion <fun> zum Debuggen
  - what: Name einer Funktion, die verfolgt werden soll
  - tracer: eine Funktion oder ein nicht ausgwerteter Ausdruck (z.B. mit **quote**/**substitute** erreichbar, vgl. Abschnitt 3.6.3 (a))
    tracer wird entweder unmittelbar vor what aufgerufen/ausgewertet, oder, sofern tracer eine Funktion ist, unmittelbar vor dem in at genannten "Schritt"
  - exit: wie tracer nur wird dies unmittelbar nach what aufgerufen/ausgewertet
  - at: (optional) Position an der tracer aufgerufen/ausgewertet werden soll
  - print: falls TRUE (default), eine erläuternde Zeile zu jeder **trace**-Auswertung wird ausgegeben

- zum Zählen der "Schritte" für Argument `at`: falls `f` die in `tracer` übergebene Funktion ist, gibt **as. list** (**body**(f)) die Schritte als Liste...
- Aufheben der Markierung durch **untrace**
- hiermit bedingtes Debuggen möglich — mit `tracer` von folgendem Typ tracer=**quote**(**if**(<condition>) <fun>)

  R–Beispiel 3.4-3 [bedingtes Debugging]:
  (aus Peng (2002))
  - bei der ML-Schätzung in einem Punktprozess taucht Funktion `nLL` auf

```
nLL ← function(mu, x) {
  z ← mu * x
  lz ← log(z)
  L1 ← sum(lz)
  LL ← mu/2−sum(lz)
}
```


- nLL macht Schwierigkeiten, sofern log auf negative Werte stößt
- Bestimmung der Nummer von "Schritt" lz ← **log**(z)

```
> as.list(body(nLL))
[[1]]
`{`
[[2]]
z ← mu * x
[[3]]
lz ← log(z)
[[4]]
L1 ← sum(lz)
[[5]]
LL ← mu/2 - sum(lz)
```

- bedingter Aufruf des Browsers, sofern in `lz` mindestens ein `NaN`

```
trace("nLL",
    quote( if(any(is.nan(lz)))
             { browser() }
         ), at=4, print=F)
```


3.4.4 Übersicht

Tabelle 3.4-4 [Tracing und Debugging in R]:

— aus Venables and Ripley (1999), Tab. 4.1

print, **cat**	manchmal genügt es, sich die wesentlichen Variablen während des Ablaufs ausgeben zu lassen
traceback	gibt die Aufrufe in Ablaufreihenfolge nach einem Abbruch aus, der einen `dump` verursacht
options`(warn=2)`	erzwingt Abbruch nach jeder Warnung
options`(error=FUN)`	spezifiziert die `dump`–Aktion, die durch einen Abbruch ausgelöst wird; default in `S-Plus` `FUN=dump.calls`, in R: `NULL`; vollständiger: `FUN=dump.frames` oder `FUN=recover`
last .**dump**	Objekt im `.Data`–directory, das alle Aufrufe/Umgebungen seit dem letzten `dump` auflistet
debugger	Funktion, die — offline — last .**dump** inspiziert
recover	ähnlich wie **debugger**, interaktives debuggen der letzten Anweisungen mit **options**(error =recover)

TABELLE 3.4-4 [TRACING UND DEBUGGING IN R — FORTSETZG.]:

browser	Funktion, die zur Unterbrechung des Ablaufs eingefügt werden kann, und mit der man dann alle aktuellen Objekte inspizieren / modifizieren kann — schon vor Auftreten eines Abbruchs
trace	spezifiziert einen Punkt, ab dem man Informationen verfolgen kann — entweder in einem Funktionskopf oder innerhalb eines Funktionskörpers; kann verwendet werden, um automatisiert **browser** einfügen zu lassen
tprint	erzeugt ein nummeriertes Listing des Funktionskörpers, das man als at Argument in **trace** verwenden kann

3.4.5 in S-Plus: inspect

In S-Plus gibt es noch zusätzlich den sehr komfortablen Befehl inspect, auf den wir hier nicht näher eingehen. Für diesen mache man sich unter der S-Plus–Hilfe **help**(inspect) schlau

3.5 Systemaufrufe

3.5.1 system

- mit dem Befehl **system** können Systemaufrufe durchgeführt werden; hängt natürlich vom Betriebssystem ab;
- Syntax: **system**(<Befehl>, intern = FALSE, wait = TRUE, input = , show.output.on.console = FALSE, minimized = FALSE, invisible = FALSE)

- Argumente
  - `<Befehl>`: (string) der auszuführende Systembefehl als String, dieser Befehl wird dabei als ein Kommando plus durch Leerzeichen getrennte Argumente geparst; falls daher der Pfad auf das Kommando Leerzeichen enthält, muss dieser in Anführungszeichen stehen
  - `intern`: (logisch) — entscheidet, ob aus der Ausgabe des Systembefehls ein R–Objekt werden soll
  - `wait`: (logisch) entscheidet, ob der R–Interpreter auf die Beendigung des Systembefehls wartet; per default wartet er; und er wartet stets, falls `intern == T`.
  - `input`: (string) wenn ein Stringvektor mit übergeben wird, wird dieser — ein String pro Zeile — auf ein `tmp`–File kopiert, und anschließend `stdin` für den Systembefehl per `pipe` auf dieses File umgelegt

- `show.output.on.console`: (logisch) entscheidet ob die Ausgabe des Systembefehls auf der R–Konsole erscheinen soll; (dies wird nicht unter `Rterm` genutzt; hier wird die Ausgabe unterdrückt, es sei denn `wait == F`)
- `minimized`: (logisch) entscheidet ob das Kommandofenster minimiert initialisiert wird
- `invisible`: (logisch) entscheidet über die Sichtbarkeit des Kommandofensters

- Rückgabewert
  - falls `intern == TRUE`: ein Stringvektor mit der Ausgabe des Kommandos, jeweils eine Zeile pro String; wenn der Systembefehl nicht ausgeführt werden kann, wird ein Fehler ausgegeben
  - falls `intern == FALSE`: ein Error Code — siehe Hilfe — es sei denn `wait == T`

  - falls `intern == FALSE` und `show.output.on.console == TRUE` erscheint die Textausgabe eines Befehls, d.h. eine Konsolen Anwendung erscheint in der R Konsole (`Rgui`) oder im Fenster, in dem R läuft (`Rterm`).

- unter `Windows`
  - der Befehl läuft direkt als ein `Windows`–Befehl unter dem `Windows API call CreateProcess`:
  - Falls keine Dateierweiterung vorliegt, werden `.exe`, `.com`, `.cmd` und `.bat` in dieser Reihenfolge ausprobiert.
  - Um `DOS`–Befehle zu nutzen, verwende man unter `Windows 9X/ME` den Systembefehl `command.com /c <Befehl>`
  - Der Suchpfad für die `command`–`Shell` hängt vom System ab; in jedem Fall enthält er das R–Verzeichnis `bin`, das Arbeitsverzeichnis und die `Windows` Systemverzeichnisse vor dem eigentlichen `PATH`.

- verschiedene Aufrufe von R
  - Was genau der Nutzer vom Systembefehl sieht, hängt davon ab, ob `Rgui` oder `Rterm` zum Aufruf von R verwendet wird:
  - Mit `Rgui` wird stets eine neue Konsole genutzt, so dass ein Kommandofenster erscheint, solange der Systembefehl arbeitet, es sei denn `invisible== T`
  - Unter `Rterm` erscheint ein separates Kommandofenster für die Konsolenanwendung nur dann, wenn `wait == F`.
- Caveat:
  - bei Aufruf kein Abbruch durch `STRG-D` mehr möglich
  - das System kann abstürzen, sofern eine Anwendung Tastatureingaben erwartet, wenn wir mit `Rgui` arbeiten und gleichzeitig `intern == T` und/oder `show.output.on.console == T` setzen.

R–Beispiel 3.5-1 [Systemaufrufe]:

```
# Aufruf eines Editors +
#     Warten bis dieser beendet ist
system("notepad myfile.txt")
# Aufruf eines Windows 9x Prozessmonitors
# (aus den Win9x KernelToys)
system("wintop", wait = F)
# Aufruf einer Shell
system("command.com")
system(paste("c:/Programme/Mozilla.org/",
             "SeaMonkey/seamonkey.exe",
             " -url cran.r-project.org",sep=""),
       wait = FALSE)
```


3.5.2 shell

- `shell` ruft einen Systembefehl auf — gewöhnlich als eine Shell
- Syntax: `shell (<Befehl>, shell, flag="/c", intern=FALSE, wait=TRUE, translate=FALSE, mustWork=FALSE, ...))`
- Argumente
  - `<Befehl>`: (string) der auszuführende Systembefehl als String
  - `shell`: (string) der Name der zu verwendenden Shell; ist dieser String `NULL`, so wird je nach Betriebssystem eine Default–Shell aufgerufen — siehe Hilfe
  - `flag`: Schalter zum Starten unter Shell; per default `"/c"`, bei `bash` oder `tcsh` `"-c"`
  - `intern`: (logisch) — entscheidet, ob aus der Ausgabe des Systembefehls ein R–Objekt werden soll

- `wait`: (logisch) entscheidet, ob der R–Interpreter auf die Beendigung des Systembefehls wartet; per default wartet er; und er wartet stets, falls `intern == T`.
- `translate`: (logisch) entscheidet ob / im `<Befehl>` in \übersetzt wird
- `mustWork`: Soll bei Misslingen des Starts des Systembefehls wird eine Fehlermeldung herausgegeben werden?
- `...`: weitere Parameter

- Rückgabewert : — wie **system**

3.5.3 Platformunabhängige Systemzugriffe

- Zeitnahme für die benötigte Zeit eines Codes
  - **system.time**
  - **proc.time**
- Umgebungsvariablen /-information

- Sys.**getenv**, Sys.putenv: inspizieren und setzen von Umgebungsvariablen
- Sys.getlocale , Sys.putlocale inspizieren und setzen der Lokaldefinition, z.B.

  "LC_COLLATE=English_United␣States.1252;LC_CTYPE=English_United␣States.1252;
  LC_MONETARY=English_United States.1252;LC_NUMERIC=C;
  LC_TIME=English_United States.1252"

- Sys.localeconv Formatierstandards für Zahlen
- Sys.**time** aktuelle Uhrzeit, Sys.timezone Zeitzone

- Filezugriff
  - **file** .access Zugriff auf Files
  - **file** .**append** Files aneinanderhängen
  - **file** .**choose** File aus einer Liste auswählen lassen
  - **file** .copy Files kopieren
  - **file** .**create** erzeugen oder abschneiden von Files
  - **file** .**exists** Test auf Existenz

- **file** .info verschiedene Informationen zu einem File
- **file** .**remove** Files löschen
- **file** .rename Files umbenennen
- **file** .**show** Darstellen eines Textfiles
- **unlink** Löschen von Files oder Verzeichnissen

- Umgang Filenamen und Pfaden
  - **basename** löst den Filenamen aus einem Pfad heraus
  - **dirname** löst den Ordnernamen aus einem Pfad heraus
  - **file** .**path** erzeugen eines Pfads aus Filename und Ordner
  - **path**.**expand** vervollständigt ~in einen Unix Pfad

3.5.4 Unix–Spezifika

- hier nicht behandelt; bei Fragen wenden Sie sich bitte an uns; im Zweifelsfall stellen wir Kontakt zu Experten her

3.6 Rekursionen und Frames

Rekursionen

- sind in R zulässig — Funktionen dürfen sich selbst aufrufen.
- sind für die Programmierung an sich schon interessant
- liefern in unserem Fall sogar besseres Verständnis für die Art und Weise, wie R Berechnungen organisiert
- sind aber oft langsam und speicherintensiv

3.6.1 Beispiel: ein adaptives Verfahren zur numerischen Integration

3.6.1 (a) Problemstellung

- gegeben eine Funktion $f : \mathbb{R} \to \mathbb{R}$, in R realisiert durch
  `f<-function(x,...){....}`
- gesucht: numerische Näherung für $I(f, a, b) := \int_a^b f(x)\, dx$ in Form einer R–Funktion
  `integriere<-function(f,a,b,...){....}`

3.6.1 (b) Lösungsansätze mit fester Gitterweite

- Trapezregel: $I^{(1)}(f, a, b) := [f(a) + f(b)](b - a)/2$
- Simpsonregel:
  $I^{(2)}(f, a, b) := [f(a) + 4f((a + b)/2) + f(b)](b - a)/6$

- Idee:
  Verwende für eine $\pm\varepsilon$ genauen Schätzung folgenden Algorithmus:
  - setze $a_{1,0} := a$, $b_{1,0} := b$, , $j := 0$ und für $i = 1, 2$: $I^{(i,0)} := I^{(i)}$
  - solange $|I^{(2,j)} - I^{(1,j)}| < \varepsilon$ wiederhole
    * $j := j + 1$
    * für $k := 1, ..., 2^{j-1}$
      spalte $[a_{k,j-1}, b_{k,j-1}]$ auf in $[a_{2k,j}; b_{2k,j}]$ und $[a_{2k+1,j}; b_{2k+1,j}]$
      mit $a_{2k,j} := a_{k,j-1}$, $b_{2k,j} := (a_{k,j-1} + b_{k,j-1})/2$
      und $a_{2k+1,j} := b_{2k,j}$, $b_{2k+1,j} := b_{k,j-1}$
    * berechne für $i = 1, 2$

$$I^{(i,j)}(f, a, b) := \sum_{k=1}^{2^j} I^{(i)}(f, a_{k,j}, b_{k,j})$$

- Nachteil: keine lokale Adaption

3.6.1 (c) Adaption durch Rekursion

(aus Venables and Ripley (1999))

Idee: Verzweigen in die Tiefe nur dann, wenn (lokal) das Abbruchkriterium noch nicht erfüllt ist

R–Beispiel 3.6-1 [numerische Integration rekursiv – I]:

```
area1 ← function(f,a,b){
d ← (a+b)/2
a1 ← (f(a)+f(b))*(b−a)/2
a2 ← (f(a)+4*f(d)+f(b))*(b−a)/6
if (abs(a1−a2)<10^(−4))
    return(a2)
area1(f,a,d)+area1(f,d,b)
}
area1(sin,0,2*pi)
```

3.6.1 (d) Feintuning

- Beschränkung der Iterationstiefe
- Angabe einer Fehlertolearanz im Argument
- zusätzliche Parameter für den Integranden
- Verwendung von **Recall** zur Ermöglichung der Kopie durch Area←area

R–Beispiel 3.6-2 [numerische Integration rekursiv – II]:

```
area ← function(f,a,b,limit=10,...,
        fa=f(a,...),fb=f(b,...),
        eps=100*Machine$double.eps^.5){
h←b−a; d←(a+b)/2; fd←f(d,...)
a1←(fa+fb)*(b−a)/2; a2←(fa+4*fd+fb)*h/6
if (abs(a1−a2)<eps) return(a2)
if (limit==0)
   {warning(paste("Maximale Rekursionstiefe",
   "erreicht nahe x=", d)); return(a2)}
Recall(f,a,d,...,fa=fa,fb=fd,
        limit=limit−1,eps=eps/2)+
Recall(f,d,b,...,fa=fd,fb=fb,
        limit=limit−1,eps=eps/2)
}
area(sin,0,2*pi)
```


3.6.2 Frames

3.6.2 (a) Definition

- ein *Frame* oder genauer *Evaluation Frame* ist eine Liste, in der Namen mit Werten verknüpft werden
- im wesentlichen zum gleichen Zweck wie Dictionaries auf der Search list
- Dictionaries und Frames werden auch als *Databases* bezeichnet

3.6.2 (b) kanonische Frames

- *local frame*: Wenn innerhalb einer Funktion oder noch allgemeiner innerhalb eines Frames ein S–Ausdruck ausgewertet werden soll, wird der Wert zu allererst aus dem aktuellen Frame bezogen; dieser heißt lokaler Frame
- alle Zuweisungen / Manipulationen geschehen, wenn nicht explizit anders verlangt (siehe Abschnitt 3.7.2(c)), im lokalen Frame

- alle Namen, die ausgewertet werden sollen, und die sich **nicht** im lokalen Frame finden, versucht man zunächst aus dem Frame des an höchster Stelle (*top level*) stehenden Ausdrucks zu beziehen; dieser Frame heißt Arbeitsverzeichnis (*working directory*) oder *frame 1*. Ist die Suche erfolgreich, wird eine lokale Kopie des Objekts angelegt.
- Wird innerhalb der Sitzung interaktiv ein Ausdruck ausgewertet, so wird zu seiner Auswertung Frame 1 initialisiert.
- Wird innerhalb Frame 1 (Frame i) eine Funktion aufgerufen, so wird zu deren Auswertung Frame 2 (Frame $(i+1)$) initialisiert.
- Findet sich der Name, der ausgewertet werden soll, weder im lokalen Frame noch in Frame 1, wird versucht, ihn aus dem Sitzungs–Frame (*session–frame*) zu beziehen; dieser Frame heißt *frame 0*.
- Findet sich der Name, der ausgewertet werden soll, weder im lokalen Frame noch in Frame 1 noch in Frame 0, so wird in der

Search list und gesucht, und falls dort nichts gefunden wird, eine Fehlermeldung ausgegeben.

3.6.2 (c) Auswertung / Manipulation von Elementen bestimmter Frames

TABELLE 3.6-3 [ZUGRIFFSFKT'S FÜR BELIEBIGE DATABASES]:

— aus Venables and Ripley (1999), Tab. 4.3

assign	erzeugt ein neues <name>=<wert>–Paar in der angegebenen Datenbank
exists	prüft ob ein gewisses Objekt in der angegebenen Datenbank zu finden ist
get	gibt eine Kopie des Objekts aus der angegebenen Datenbank zurück, falls es existiert; sonst einen Fehler
objects	gibt einen Stringvektor mit den Namen der Objekte der angegebenen Datenbank zurück
remove	löscht ein Objekt aus der angegebenen Datenbank

- alle Funktionen haben als Argument `envir`;
- envir=**sys.frame**(n) $\rightsquigarrow$ Frame Nummer n spezifizieren
- alternativ dazu `pos`, um eine Position innerhalb der Suchliste (Search list) anzugeben
- **get** und **exists** durchsuchen per default den gesamten Suchpfad
- die anderen per default den lokalen Frame

Tabelle 3.6-4 [weitere Database–Zugriffsfunktionen]:

←	macht eine Zuweisung im Arbeitsverzeichnis
sys.parent(n)	gibt die Nummer des Ahnen n–ter Ordnung des lokalen Frames aus
ähnliches	sys.function, sys.call, sys.calls, sys.frame, sys.frames, sys.nframe, sys.parent, sys.parents, parent.frame, sys.on.exit, sys.status, vgl. Online–Hilfe zu R

3.6.2 (d) Beispiel für die Kommunikation zwischen Frames

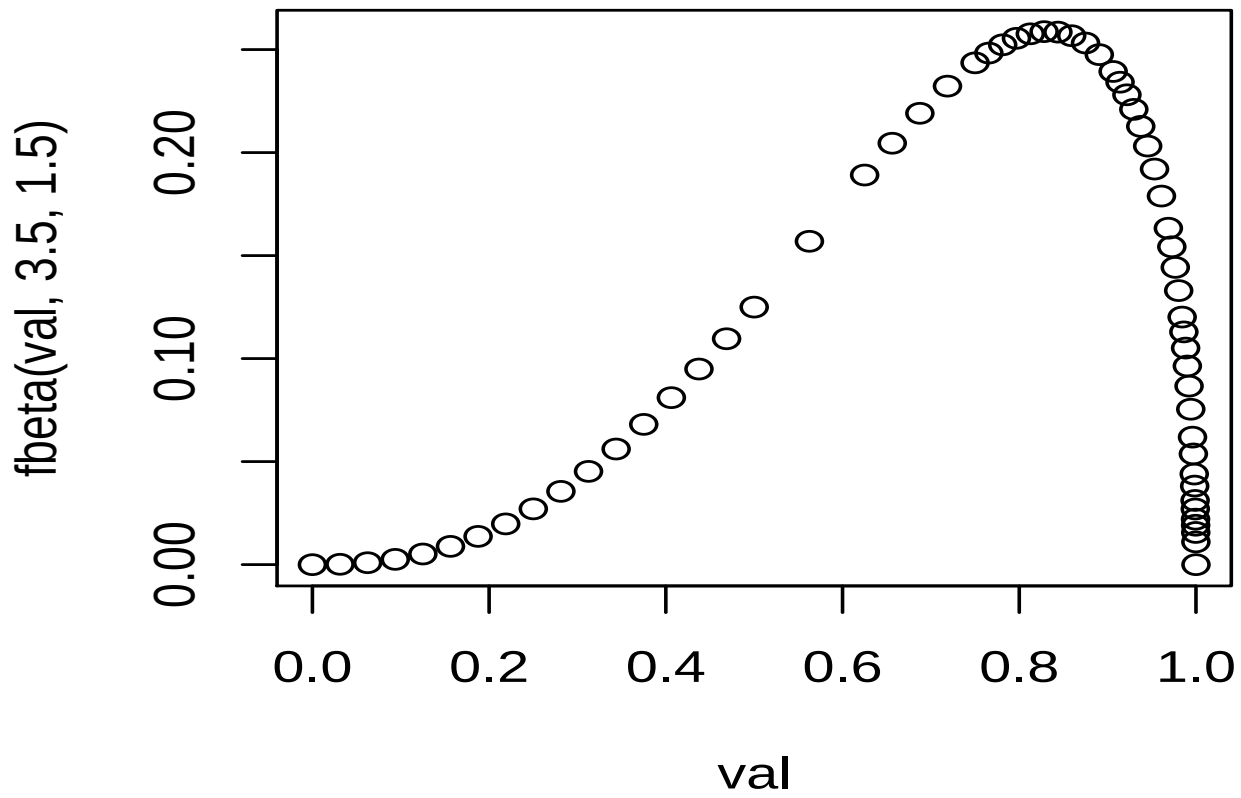
R–Beispiel 3.6-5 [Plot der Auswertungsstellen von area]:

```
# Ziel: Ausgabe aller Auswertungsstellen
#       von area aus R–Beispiel 3.6–2
# Variable val in frame 0 erzeugen
```

area aus R–Beispiel 3.6-2

```
assign("val",NULL,envir=sys.frame(0))
  # Erzeugung der ursprgl. Funktion
fbeta<-function(x,alpha,beta){
  x^(alpha-1)*(1-x)^(beta-1)}

  # Erzeugung der Funktion, die
  # ihre Auswertungsstellen protokolliert
fbeta.tmp<-function(x,alpha,beta){
  assign("val",c(val,x),envir=sys.frame(0))
  x^(alpha-1)*(1-x)^(beta-1)}
  # Ausintegrieren mit area
b0<-area(fbeta.tmp,a=0,b=1,limit=12,
         alpha=3.5,beta=1.5,eps=10^(-3))
b0
  # ploten der Funktion an den
  # mitprotokollierten Auswertungsstellen
plot(val,fbeta(val,3.5,1.5))
```


Auswertungsstellen in `area`



3.6.3 Programmieroperationen auf der Sprache

vgl. R Language Definition (2006c), Kapitel 6

3.6.3 (a) Calls und Expressions

- ein *Call* ist so etwas wie ein schon syntaktisch aufgelöster, aber noch nicht ausgewerteter Ausdruck
- eine *Expression* umfasst einen oder mehrere geparste aber unausgewertete *Ausdrücke*, wobei
- ein *Ausdruck* eine syntaktisch korrekte *Tokenkette* ist und
- eine *Tokenkette* eine Anreihung von *Tokens* und
- ein *Token* ein (terminales) Schlüsselwort der Sprache S ist
- z.B. erhält man mit u←**quote**(**plot**(x, **sin**(**tan**(y)), u, v)) einen Call;
  - ist im wesentlichen eine Liste (Umwandlung hin und her mit **as.list**, **call** möglich

  - erstes Listenelement: Name der zu äußerst stehende Funktion
  - weitere Listenelemente: die Argumente dieser Funktion
  - ein weiteres Parsing in Unterausdrücke findet nicht statt — **sin**(**tan**(y)) wird nicht aufgelöst
  - die entsprechende Expression wäre weiter aufgelöst
- Expressions und Calls werden mit **eval** ausgewertet

3.6.3 (b) Parsing und Deparsing

- um einen Call/eine Expression zu parsen ohne dass ausgewertet wird, verwendet man **quote** oder **substitute**
- um von einem Call/einer Expression wieder zum Klartext zu kommen, verwendet man **deparse**

- im Zusammenhang mit Lazy evaluation, vgl. Abschnitt 3.3.1 (c) nützlich:
  - am Beginn der Standardplotfunktion:
    xlabel ← **if** (**!missing**(x)) **deparse**(**substitute**(x))
  - x ist noch nicht ausgewertet; **substitute** verhindert die Auswertung und **deparse** rekonstruiert den Klartext von Argument x; damit erhält man eine schöne Achsenbeschriftung
- mehrfach verwendeter Code-Block wird zweimal verwendet
  - Vorteil: Code ist nur an einer Stelle zu pflegen
  - Unterschied zu einer Funktion: kein environment wird angelegt

R–Beispiel 3.6-6 [Code-Block als Call in Variable]:

* als Call:

```
fu ← function(x,y)
 {A=quote({ # Hier faengt ein mehrfach
            # replizierter Code an
          f=sin(x);   m=cos(y);   z=min(x,f,m)
          mi=abs(round(10*m,0))
          print(mi)
          mm=matrix(z*rnorm(mi*3,0,2),mi,3)
          print(mm)
          })
  eval(A)
  print(c(mode(A),environment(A)))
 #....
 #hier kaeme nun irgendwas

  eval(A)
}
```


* als Funktion ohne Argumente (geht wegen Lexical Scoping):

```
##subtil anders als
  fu2 <- function(x,y)
   {A=function(){
      # Hier faengt ein mehrfach
      # replizierter Code an
           f=sin(x);   m=cos(y);   z=min(x,f,m)
           mi=abs(round(10*m,0))
           print(mi)
           mm=matrix(z*rnorm(mi*3,0,2),mi,3)
           print(mm)
      }
    A()
    print(c(mode(A),environment(A)))
    #....
    #hier kaeme nun irgendwas

    A()
```


Peter Ruckdeschel
Matthias Kohl

```
}
```

* der Vergleich

```
> fu(3,4)
[1] 7
           [,1]       [,2]        [,3]
[1,] -1.1563608  0.5779231 -0.01568853
.....
[7,]  0.8918121  0.8003143 -1.27947251
[1] "call"
[1] 7
            [,1]        [,2]       [,3]
[1,] -0.74909778 -1.11834236 -2.5217689
.....
[7,]  0.31102524  3.12462180  0.1319528
> fu2(3,4)
[1] 7
              [,1]       [,2]       [,3]
[1,] -1.707287320  1.5503889 -0.4610366
.....
[7,]  1.034514241  0.7733199  0.6844463
[[1]]
[1] "function"

[[2]]
<environment: 0x02273acc>

[1] 7
           [,1]       [,2]       [,3]
```


```
[1,]   1.2114287   1.0617850   2.6441793
.....
[7,]  -0.8499366  -0.1239640  -0.4150593
```


4 Graphiken

4.1 Ausgabegeräte

- Vor der Anfertigung von Grafiken müssen wir erst einmal spezifizieren, auf welchem Gerät bzw. in welchem Format wir die Grafik erzeugen wollen; $\rightsquigarrow$ Ausgabegerät oder *Devices*
- Angabe aller verfügbaren Devices durch ?Devices

4.1.1 Betriebssystem–Treiber

je nach Betriebssystem wird ein Grafikfenster mit unterschiedlichen Befehlen erzeugt;

Tabelle 4.1-1 [Grafik–Devices für Betriebssysteme]:

windows	unter `Windows`
X11	unter `Unix`
macintosh	unter `Macintosh`

- hier: nur `Windows`–Welt
- `windows` startet ein neues Grafikfenster und lenkt alle folgenden Grafikausgaben auf dieses Fenster
- `win.graph`, **`x11`**, **`X11`** sind (unter `Windows`) Synonyme — aus Kompatibilitätsgründen!
- Syntax:
```
windows(width=7, height=7, pointsize=12,
record = getOption("graphics.record"), rescale =
c("R", "fit", "fixed"), xpinch, ypinch, canvas =
"white", gamma = getOption("gamma"))
```
- Argumente
  - `width`, `height`: (nominale) Breite / Höhe des Grafikfensters in Zoll
  - `pointsize`: die default–Schriftgröße in Punkt
  - `record`: (logisch): setzt den Anfangszustand der Flagge, die das Aufzeichnen von Plots regelt — siehe Online–Hilfe
  - `rescale`: kontrolliert, was bei Umskalierungen von Plots passieren soll — siehe Online–Hilfe

- `xpinch`, `ypinch`: Pixel pro Zoll, horizontal und vertikal
- `canvas`: (color) Farbe des Hintergrundes, falls keine Hintergrundfarbe explizit angegeben ist
- `gamma`: Gamma–Korrekturfaktor

- Details
  - die Fenstergröße wird per default aus oft unzuverlässigen `Windows`–Angaben über das Display bestimmt;
    stattdessen auch möglich: Angabe von `xpinch` und `ypinch`
  - bei Aufruf: Grafikfenster nicht größer als 85% der Höhe oder Breite des Bildschirms;
    Höhe und Breite werden bei Bedarf proportional umskaliert;
    nachträglich: interaktiv Größe des Grafikfensters noch vergrößerbar;
  - nach Umskalierungen wird Grafikfenster per default neu gezeichnet;
    Option `"fit"`: Plot wird auf die neue Größe des Grafikfensters umskaliert;

Option "fixed": Dimensionen der Grafik unverändert; Rollbalken hinzugefügt

- Fläche außerhalb des Grafikfensters in Windows application background colour; Gebiet des Grafikfensters in Farbe aus canvas, es sei denn Hintergrundfarbe angegeben
- Unterschied zwischen canvas und background colour: erstere wird beim Kopieren nicht mitkopiert, letztere schon
- aufgenommene plot histories sind von der Klasse "SavedPlots"; sie haben eine print– und eine subset–Methode;
- einzelne aufgenommene Plots sind aus Klasse "recordedplot" $\rightsquigarrow$ können durch print neu ausgegeben werden;

- Rückgabewert: keiner; es wird ein Fenster geöffnet

4.1.2 Ausdruck mit postscript

4.1.2 (a) postscript

- startet einen Grafiktreiber zur Erzeugung von Postscript–Files und lenkt alle Grafikausgaben auf diesen Treiber
- Syntax: **postscript** `(file = ifelse(onefile, "Rplots.ps", "Rplot%03d.ps"), onefile = TRUE, paper, family, encoding, bg, fg, width, height, horizontal, pointsize, pagecentre, print.it, command)`
- Argumente
  - `file`: (string): Name des zu erzeugenden Postscript–Files
  - `onefile`: (logisch): Sollen mehrere Seiten in ein File geschrieben werden?
  - `paper`: Papierformat im Drucker; zur Auswahl stehen: `"a4"`, `"letter"`, `"legal"` und `"executive"`; auch: `"special"`, sofern `width` und `height` das Papierformat spezifizieren

- `family`: Schriftenfamilie; genaueres siehe Online–Hilfe
- `encoding`: Schriftenkodierungsfile; per default `"R_HOME/afm/WinAnsi.enc"`; genaueres siehe Online–Hilfe
- `bg`, `fg`: (color) — default–Werte für die Vorder– (`fg`) und Hinter–(`bg`)grundfarbe
- `width`, `height`, `pointsize`: wie bei windows
- `horizontal`: (logisch) `TRUE`$\hat{=}$quer, `FALSE`$\hat{=}$hochkant
- `pagecenter`: (logisch) Soll die Seite auf dem Papier zentriert werden?
- `print.it`: (logisch) Soll das File anschließend gleich an den Drucker gesandt werden?
- `command`: (string): auszuführender Befehl zum Ausdruck des `Postscript`–Files

4.1.2 (b) ps.options

- setzt die Optionen des `Postscript`–Treibers
- Syntax: **ps.options**`(paper, horizontal, width, height, family, encoding, pointsize, onefile = TRUE, print.it = FALSE, bg, fg, append = FALSE, reset = FALSE, override.check = FALSE)`
- Ausgabe mit `.PostScript.Options`
- Argumente
  - `paper`, `horizontal`, `width`, `height`,`family`, `encoding`, `pointsize`, `bg`, `fg`, `onefile`, `print.it` wie bei **postscript**
  - `append`: (logisch): nur aus Kompatibilitätsgründen
  - `reset`,`override.check`: (logisch) werden an **check.options** weitergeleitet — siehe Online–Hilfe

4.1.3 andere Ausgabeformate

Neben dem **postscript** – und den systemabhängigen Grafikfensterbefehlen gibt es noch weitere Formate auf die man in R Grafik schreiben kann

TABELLE 4.1-2 [WEITERE GRAFIK–TREIBER]:

pdf	schreibt PDF–Format auf ein File
pictex	schreibt LaTeX/PicTeX–Format auf ein File
windows	neben Bildschirmtreiber auch WMF–Format auf File erzeugbar
png	schreibt PNG–Format auf ein File
jpeg	schreibt JPEG–Format auf ein File
bmp	schreibt BMP–Format auf ein File
xfig	schreibt XFIG–Format auf ein File

BEMERKUNG 4.1-3:

Schließen aller (oder einzelner) Graphikfenster / –Files mit

graphics . off

4.2 der `plot` und der `par` Befehl

nach einem Referat von *Matthias Brandl* vom 03.06.2002

4.2.1 Die par – Funktion

- setzt diverse Graphikoptionen fest, so dass alle nachfolgenden Graphikbefehle diese Optionen verwenden
- Um die verschiedenen Optionen, die mit **par** gesetzt werden können, zu sehen, tippe man am Prompt ein
  `> par()`
  - $\rightsquigarrow$ 60 Optionen für **par**
  - Hier sieht man auch, wie diese Optionen voreingestellt sind (So werden sie auch nach einem Neustart von R wieder verwendet!).
  - für eine detailliertere Beschreibung $\rightsquigarrow$ `>?par`

- häufigste Parameter/Argumente — mit Voreinstellungen

`fig`	Koordinaten der gesamten Einzelabbildung
`fin=c(m,n)`	Größe der Einzelabbildung in Inch. $m\hat{=}$ Breite, $n\hat{=}$Höhe
`pin=c(m,n)`	Größe der Graphik in Inch. s.o.
`mar=c(5,4,4,2)+0.1`	Alle Ränder in Zeilenanzahl
`mai=c(1.4,1.1,1.1,0.6)`	Alle Ränder in Inch
`oma=c(0,0,0,0)`	Äußere Randlinien
`omi=c(0,0,0,0)`	Dicke der äußeren Ränder in Inch
`plt=c(0.1,0.9,0.2,0.9)`	Koordinaten der Plotregion als ein Teil der Einzelabbildung
`usr`	Minimum und Maximum für x- und y-Achse

`mfrow=c(m,n)`	Multiples Abbildungslayout; zeilenweises Plotten, erzeugt eine (m, n)-Matrix von Einzelabbildungen
`mfcol=c(m,n)`	s.o.; spaltenweise

4.2.2 Befehle zur Aufteilung des Graphsheets

layout`(mat,`	teilt das Graphsheet in so viele
`widths=rep(1,dim(mat)[2]),`	Zeilen und Spalten wie die Matrix
`heights=rep(1,dim(mat)[1]))`	`mat` Zeilen und Spalten hat
	`widths` legt die Breite der einzelnen Zellen, `heights` deren Höhe fest
split .**screen**`(figs = c(m,n))`	Teilt das Graphsheet in verschiedene Screens auf (vgl. mfrow)
screen`(n)`	Aktiviert den Screen Nummer `n`

4.2.3 Die plot–Funktion

- ist die elementarste Funktion, um eine Abbildung zu erzeugen
- Syntax: **plot**`(x,y, <Optionen>)`
- mit `type=` kann man die Art des Datenauftrags verändern:

`type="p"`	Punkte (Voreinstellung)
`type="l"`	die Daten sind durch Strecken verbunden
`type="b"`	beides (Punkte mit Strecken)
`type="h"`	vertikale Stäbe
`type="o"`	Strecken mit darüberliegenden Punkten
`type="s"`	Treppenstufen
`type="n"`	nichts

- Modifizierende Optionen (viele auch in **par()** einstellbar):

`axes`	(logisch) mit / ohne Achsen
`main`	(String) Titel
`sub`	(String) Untertitel
`xlab`	Beschriftung der x-Achse
`ylab`	Beschriftung der y-Achse
`xlim=c(xmin,xmax)`	linke und rechte Grenze der x-Achse
`ylim=c(ymin,ymax)`	untere und obere Grenze der y-Achse

- Zeichentypus

`pch = "*"`	spezielles Plotzeichen; auch als Vektor der Länge von `x` übergebbar
`lwd = 1`	Linienbreite; auch als Vektor der Länge von `x` übergebbar
`lty = 1`	Linientyp ($1 \hat{=}$ durchgezogen, $2 \hat{=}$ gestrichelt, ...); auch als Vektor der Länge von `x` übergebbar
`col = 1`	Farbe (systemabhängig); in R unter Windows: auch (englische) Farbnamen als String übergebbar; auch als Vektor der Länge von `x` von Farbcodes/–namen übergebbar
`box`	(logisch) zeige oder verberge den Rahmen um die Abbildung

- Befehle, um Linien zu einem Graph hinzuzufügen:

abline(a,b)	Fügt eine Gerade mit y-Abschnitt a und Steigung b hinzu
abline(h)	Fügt eine horizontale Gerade auf der Höhe h hinzu
abline(v)	das Analogon für vertikale Geraden
arrows(x1,y1,x2,y2)	Fügt einen Pfeil von (x1,y1) nach (x2,y2) hinzu
box()	Fügt den äußeren Rahmen hinzu
lines(x,y)	Fügt eine Gerade hinzu
points(x,y)	Fügt einen Punkt hinzu
segments(x1,y1,x2,y2)	Fügt eine Strecke von (x1,y1) nach (x2,y2) hinzu

- Befehle, um Text zu einem Graph hinzuzufügen — der Befehl **text**
  - Syntax: **text**`(xpos,ypos,<text>,adj=0.5,cex=1, col=1,crt=0,srt=0,font=1)`
  - Fügt Text `<text>` an einer definierten Stelle hinzu
  - Argumente

`xpos,ypos`	x- und y-Koordinate des Textes im Bild
`adj=0.5`	Textausrichtung; $0 \mathrel{\hat{=}}$ linksbündig, $0.5 \mathrel{\hat{=}}$ zentriert, $1 \mathrel{\hat{=}}$ rechtsbündig
`cex=1`	Fontgröße
`col=1`	Farbe
`crt=0`	Rotation eines Zeichens in Grad (im Uhrzeigersinn von der Horizontalen weg)
`srt=0`	Rotation der Zeichenkette
`font=1`	Font (systemabhängig)

- weitere Modifikatoren als separate Befehle

mtext`(....)`	Text in einem Rand der Abbildung
title `("Titel", "Untertitel")`	Fügt Titel und/oder Untertitel hinzu
`axes()`	Fügt x- und y-Achse hinzu

- Achsenspezifikation: der Befehl **axis**
  - Syntax:

```
axis(side=n,at=x,labels=s,pos=y,las=m,mgp=c(3,1,0),
    xaxt="....",yaxt="....",tck=-0.02,lty=1,lwd=1)
```

  - Fügt eine spezielle Achse hinzu
  - Argumente

`side=n`	$n = 1$: x-Achse, $n = 2$: y-Achse,....
`at=x,labels=s`	schreibt die Beschriftung `s` an die Stelle `x`
`pos=y`	verschiebt die Achse bis sie durch die Koordinate `y` geht

`las=m`	$m = 0$ fügt die Beschriftung parallel, $m = 1$ horizontal und $m = 2$ um 45 Grad gedreht zur Achse hinzu
`mgp=c(3,1,0)`	Randlinie, an der Achsentitel, Beschriftung und die Linie selbst angebracht werden
`xaxt="...."`	x-Achsen-Typ (vgl. type="...")
`yaxt="...."`	y-Achsen-Typ
`tck=-0.02`	Länge der Markierungsstriche
`lty=1`	Linientyp (s.o.)
`lwd=1`	Linienbreite

4.2.4 alternatives Paket zu plot: grid

Paul Murrell hat alternativ zu der bisher dargestellten Graphik ein eigenes Paket **grid** zur Verfügung gestellt, das weitaus leistungsfähiger ist als die hier dargestellten Befehle; näheres siehe **library(grid, help)**

4.2.5 ein Beispiel

```
data(geyser, package=KernSmooth)
attach(geyser)
geyser.both←cbind(waiting, duration)
layout(mat=matrix(c(1,2,4,3),ncol=2),
       widths=c(0.7,0.3),heights=c(0.3,0.7))
par(mar=c(2,4,2,2), cex=0.7)
hist(geyser$waiting)
par(mar=c(4,4,2,2), cex=0.7)
plot(geyser$waiting, geyser$duration, pch="*",
     xlab="Wartezeit", ylab="Dauer der Eruption")
title("\nOld Faithful Geyser Data Set", cex=0.5)
par(mar=c(4,3,2,2), cex=0.7)
boxplot(geyser$duration)
```

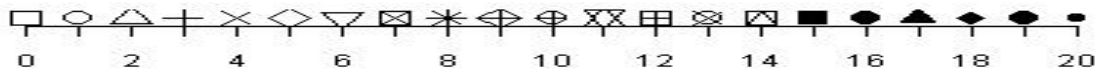
---

zum Abschluß **demo(graphics)**

4.3 einige Tabellen

4.3.1 Symbole für pch

Angabe einer Zahl für pch

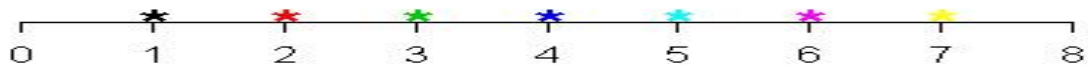


BEMERKUNG 4.3-1:

Bei Angabe numerischer Codes für pch Steuerung der Größe durch mkh

4.3.2 Farben

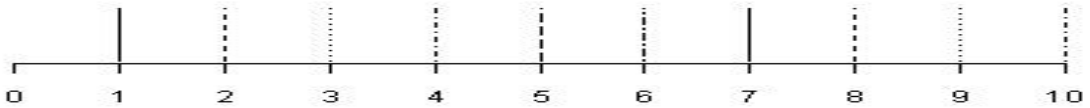
Angabe einer Zahl für **col**



- Auflistung aller (englischer) Farb*namen*, die statt numerischer Werte verwendbar sind, mit **colors**() — zur Zeit 657 Namen!
- mit **palette** kann man eine neue Palette an Farben definieren, die die numerischen Werte überschreibt
- vordefinierte Skalen / "Topologien"
  - **rgb**: Farbkoordinaten im `rgb`–Raum (*rot*, *grün*, *blau*), codiert in $[0,1]^3$
  - **hsv**: Farbkoordinaten im `HSV`–Raum (*hue*, *saturation*, *value*), codiert in $[0,1]^3$
  - **gray** (Synonyom: **grey**): Grauskala, codiert in $[0,1]$, $0\hat{=}$ schwarz, $1\hat{=}$ weiß
  - weitere: **rainbow** , **heat.colors**, **topo.colors**, **terrain.colors**, **cm.colors** (siehe Online–Hilfe)
- Umsetzung der Farbnamen / –nummern (einer Palette) / –skalen in `rgb`–Koordinaten durch col2rgb
- konzeptionelle Wahl der Farben: vgl. `http://colorbrewer.org/`; in R umgesetzt im Paket RColorBrewer von Erich Neuwirth

4.3.3 Linientypen

Angabe einer Zahl für lty

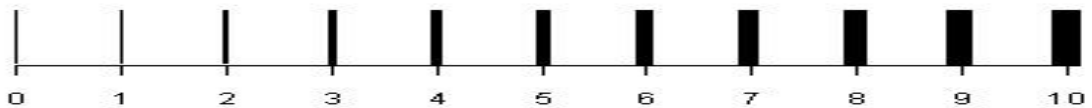


genauer:

- eine der drei Möglichkeiten
  - Zahl (wie oben)
  - Name (`character string`) aus `"blank"`, `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, `"longdash"`, `"twodash"`
  - eine Zeichenkette geradzahliger Länge mit bis zu 8 Zeichen aus `c(1:9,"A":"F")` (hexadezimal codiert) — abwechselnd Zahl der gezeichnet und der nicht gezeichneten Einheiten, z.B. ergibt `A24C33` Muster `10+2-4+12-3+3-` (+0$\hat{=}$ zeichnen, -0$\hat{=}$ nicht zeichnen)

4.3.4 Linienbreiten

Angabe einer Zahl für lwd



BEMERKUNG 4.3-2:

Referenz zu graphischer Visualisierung von Daten: Cleveland (1985)

4.4 weitere grundlegende Plot–Befehle

4.4.1 eine Übersicht

TABELLE 4.4-1 [GRUNDLEGENDE GRAFIKBEFEHLE]:

— aus Venables and Ripley (1999), Tab. 3.1

Befehl	c.f.	Zweck
abline	4.2	fügt eine Linie in die aktuelle Grafik
barplot	4.4.2 (b)	Säulendiagramm
biplot	4.4.3 (c)	gemeinsame Darstellung von Zeilen– und Spaltenraum einer multivariaten Variable in einem Plot^{S+}
brush, spin	4.4.3 (c)	dynamische Grafik^{S+}
contour	4.4.6	Niveaulinien–Plot
coplot	4.6	plotten einer Variablen gegen eine andere zu gegebenen Wertbereichen oder Niveaus einer der beiden Variablen

$\cdot^{S+} \hat{=}$ nur in S-Plus

Tabelle 4.4-1 [grundlegende Grafikbefehle]:

Befehl	c.f.	Zweck
dotchart	4.4.2 (c)	erzeugt ein Punkt–Diagramm
faces	4.4.3 (c)	Darstellung multivariater Daten in Chernoff–Gesichtern[s+]
frame	4.5.1 (a)	weitergehen zur nächsten Abbildungs–Region
hist	2.5.1	Histogramm (verwendet **barplot**)
hist2d	4.4.3 (c)	zweidim. Histogramm[s+]
identify , **locator**	4.5.3	Interaktion mit existierendem Plot
image	4.4.6	"High–Density"–Plot
legend	4.5.2 (c)	fügt eine Legende hinzu
matplot	4.4.3 (c)	simultanes Plotten mehrerer Kurven: jedem x-Wert wird für jede Spalte einer Matrix ein Punkt als y–Wert abgetragen

Tabelle 4.4-1 [grundlegende Grafikbefehle]:

Befehl	c.f.	Zweck
mtext	4.2	fügt Text in den Rand ein
pairs	4.4.3 (a)	eine Ansammlung aller paarweisen Plots einer multivariaten Variable
par	4.2	setzen / inspizieren von Graphikparametern
persp, perspp^{s+}	4.4.6	3D–Plot–Befehle
pie	4.4.2 (d)	Tortendiagramm
plot	4.2	generischer (vgl. Abschnitt 8.1.5 (b)) Plotbefehl
polygon	4.2	fügt ein Polygon in ein bestehendes Diagramm ein
points, **lines**	4.2	fügt Punkte oder Linien in ein bestehendes Diagramm ein
qqplot, **qqnorm**	5.1.8	Quantil–Quantil–Plot und normaler Q-Q–Plot
segments	4.2	fügt Liniensegmente oder Pfeile in ein
arrows		bestehendes Diagramm ein

Tabelle 4.4-1 [grundlegende Grafikbefehle]:

Befehl	c.f.	Zweck
stars	4.4.3 (c)	Darstellung multivariater Daten in Star–Plots
symbols	4.4.3 (c)	fügt Symbole von variierender Größe in ein bestehendes Diagramm ein
text	4.2	fügt Text in ein bestehendes Diagramm ein
title	4.2	fügt einen Titel in ein bestehendes Diagramm ein

4.4.2 Univariate Graphiken

4.4.2 (a) Histogramme, emp. Verteilungsfunktion, Boxplots

siehe Abschnitte 2.5.1, 2.5.2

zusätzlich: z.B. violinplot in Paket `UsingR`

4.4.2 (b) Säulendiagramme — barplot

- Syntax:

```
barplot(height, width = 1, space = NULL, names.arg = NULL,
        legend.text = NULL, beside = FALSE, horiz = FALSE,
        density = NULL, angle = 45, col = heat.colors(NR),
        border = par("fg"), main = NULL, sub = NULL,
        xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
        xpd = TRUE, axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names =
        par("cex.axis"), inside = TRUE, plot = TRUE, ...)
```

- Argumente
  - `height`: entweder ein Vektor oder eine Matrix von Werten, die die Säulen aus dem Plot beschreiben.
    - falls `height` ein Vektor ist:
      eine Folge rechteckiger Säulen mit Höhen entsprechend `height` wird geplottet
    - falls `height` eine Matrix und `beside FALSE` ist:
      Jede Säule besteht aus Teilsäulen, entsprechend den Werten in

den Spalten von `height`

  * falls `height` eine Matrix und `beside TRUE` ist:
    Die einzelnen Spaltenelemente werden nebeneinander geklebt.

- `width`: optionaler Vektor mit Säulenbreiten. Angabe eines einzelnen Wertes wirkt sich nur aus, sofern `xlim` spezifiziert ist.
- `space`: leerer Zwischenraum links vor einer Säule, gemessen als Anteil der durchschnittlichen Säulenbreite;
  angebbar als einzelner Wert oder je ein Wert / Säule;
  falls `height` als Matrix vorliegt und `beside TRUE` ist, kann `space` als jeweils zwei Werte angegeben werden; der erste ist dann der Zwischenraum zwischen zwei Säulen einer Spalte, der zweite dann zwischen den Spalten; in diesem Fall ist der default `c(0,1)` sonst `0.2`.
- `names.arg`: ein Stringvektor mit Namen, die unter die Säulen(gruppen) geplottet werden; per default werden hier die (Spalten)Namen vom `names`–Attribut von `height` übernommen;
- `legend.text`: ein Stringvektor, um eine Legende für den Plot zu erstellen, oder ein logischer Wert, der bestimmt, ob eine Legende

eingeschlossen werden soll; oft nützlich wenn `height` eine Matrix ist; in diesem Fall entsprechen die Zeilen von `height` den in der Legende aufgeführten Labels; falls `legend.text TRUE` ist werden per default die Zeilennamen von `height` verwendet.

- `beside`: (logisch) siehe `height`
- `horiz`: (logisch) per default `FALSE`; sollen die Säulen horizontal statt vertikal geplottet werden?
- `density`: ein Vektor der die Dichte der Schraffur der Säulen(komponenten) in Linien/Zoll angibt, per default `NULL`; dann keine Schraffur
- `angle`: Neigung der Schraffur als Winkel im Gegenuhrzeigersinn
- `col`: (Vektor)(color) Farben der Säulen(komponenten)
- `border`: Randfarbe der Säulen.
- `main`, `sub`, item `xlab`, `ylab`, `xlim`, `ylim`, `axes`, `axisnames`, : wie bei **plot**
- `xpd`: (logisch) dürfen Säulen aus dem Bild ragen?
- `cex.axis`, `cex.names`: (`cex`$\hat{=}$ *c*haracter *ex*pansion)

Vergrößerungsfaktor für die Achsenbeschriftung / Namen.

- `inside`: (logisch) sollen Linien, die zwei aneinander angrenzende Säulen trennen, mitgezeichnet werden?
- `plot`: (logisch) soll überhaupt etwas geplottet werden — oder ist man nur auf den Rückgabewert aus?
- `...`: weitere Graphikparameter (**par**), die an **plot.window()**, **title ()** und **axis** weitergeleitet werden.

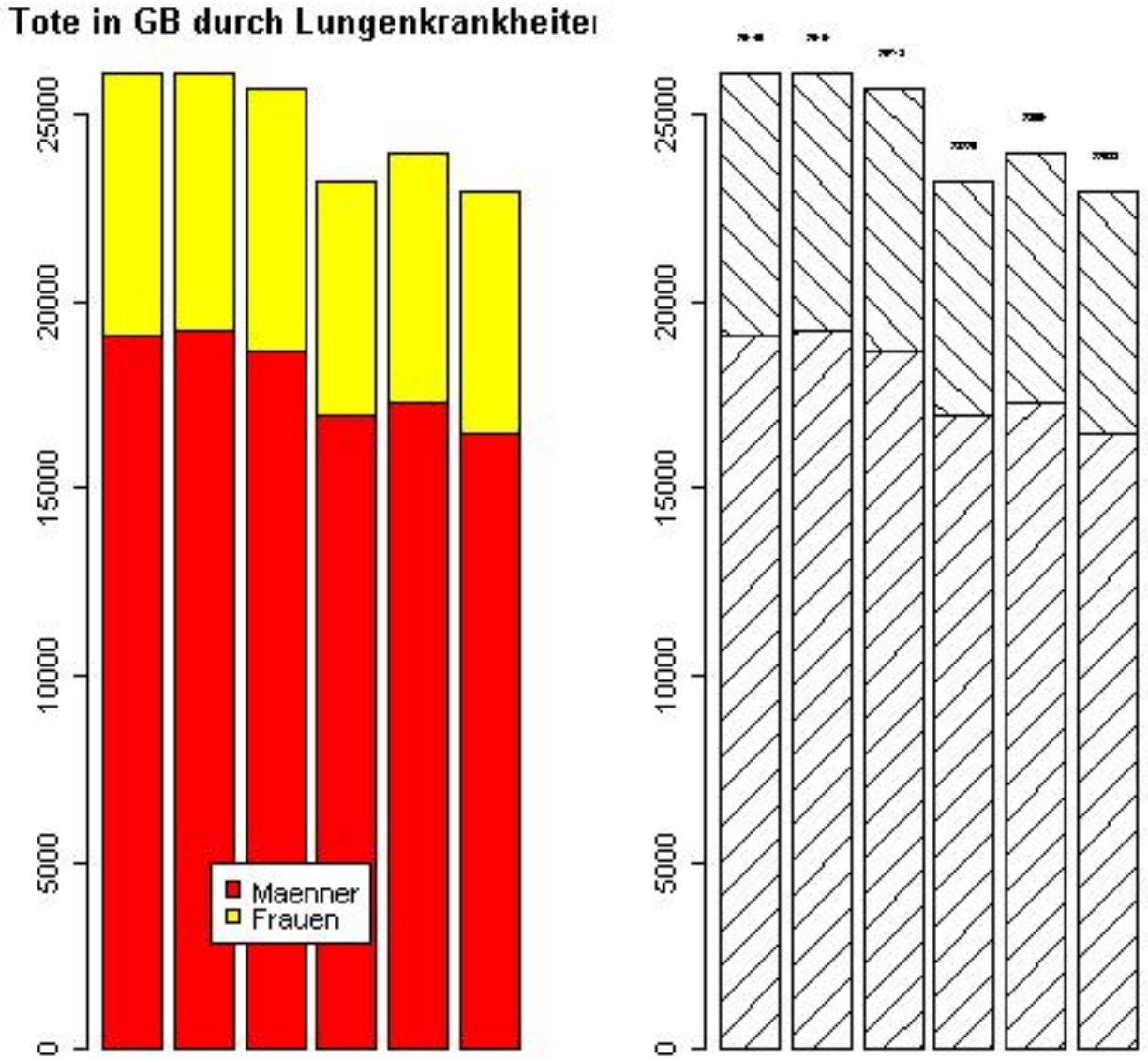
- Rückgabewert
  - ein numerischer Vektor (oder Matrix), sofern `beside = TRUE`, ergibt die x–Koordinaten aller Säulenmitten — nützlich um in diesen Plot weitere Information zu plotten.

R–Beispiel 4.4-2 [Tote in GB durch Lungenkrankheiten]:

```
#Einladen notwendiger Bibliotheken
library(MASS)
library(lattice)
library(ts)
#Bereitstellen der Datensaetze
data(mdeaths); data(fdeaths);
# (M aenner und F rauen)
lungen.tote ← aggregate(
        ts.union(mdeaths, fdeaths), 1)
# Aggregieren der Daten in eine Matrix
# mit Spalten m / w  und Zeilen Jahr
# 1. Saeulendiagramm
barplot(t(lungen.tote), names =
   dimnames(lungen.tote)[[1]],
   main = "Tote␣in␣GB␣durch␣Lungenkrankheiten")
```


```
# interaktives Festlegen der Position
# der Legende --
if(interactive())
    legend(locator(1), c("Maenner", "Frauen"),
           fill = c(2, 3))
# 2. Sauelendiagramm
#    - ablegen der x-Positionen in loc
loc <- barplot(t(lungen.tote), names =
    dimnames(lungen.tote)[[1]], angle = c(45, 135),
    density = 10, col = 1)
```


```
# Berechnung der absoluten Zahlen der
# Lungentoten fuer jedes Jahr
total ← rowSums(lung.deaths)
# Beschriftung des Saeulen mit diesen
# Zahlen oberhalb der Saeulen
# mit entsprechendem Abstand
text(loc, total + par("cxy")[2], total,
     cex = 0.7, xpd = T)
```

Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

4.4.2 (c) Punktdiagramme — dotchart

- Alternative zu Säulendiagrammen; erstellt zwei Varianten von Dot–Plots wie sie sich in Cleveland (1985) finden
- auch für gruppierte Variablen — siehe Beispiel in session8.r
- Syntax:

```
dotchart(x, labels = NULL, groups = NULL, gdata = NULL,
         bg = par("bg"), cex = par("cex"), pch = 21,
         gpch = 21, color = par("fg"), gcolor = par("fg"),
         lcolor="gray",main=NULL,xlab=NULL,ylab=NULL,...)
```

- Argumente
  - `labels`: String–Vektor mit Namen für jeden Punkt
  - `groups`: Gruppeneinteilung
  - `gdata`: Gruppen–Daten; normalerweise **summary**–Werte
  - `gpch`: Symbol(e) für die Gruppenvariablen

- `gcolor`, `lcolor`: Gruppen–/ Labelfarbe
- Rest wie **barplot**

4.4.2 (d) Tortendiagramme — pie

- in R: pie
- Zitat aus der R–Hilfe

  Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.

- daher hier nicht ausführlich behandelt
- siehe Beispiel in session8.r

4.4.3 multivariate Diagramme

- bisher maximal Paare von Zufallsvariablen auf einmal betrachtet
- $\rightsquigarrow$ Wie visualisiert man multivariate Daten?

4.4.3 (a) Scatterplots

- in einem matrix–artigen Schema wird jede Variable gegen jede abgetragen
- in R: **pairs**

R–Beispiel 4.4-3 [Fertilität und soz.ök. Faktoren in CH]:

```
#Einladen notwendiger Bibliotheken
library(MASS); library(modreg); data(swiss)
#Info ueber Datensatz
help(swiss)
```


```
#Wandeln in Data.frame
swiss.df<-data.frame(
      Fertility=swiss[,1],swiss[,-1])
#Funktion zum simultanen Plotten der Punkte
# und einer lokalen robusten Regression
points.lines<-function(x,y,...)
  {points(x,y,...);
   lines(loess.smooth(x,y),...)}
#Erzeugen des Scatterplots
pairs(swiss.df,panel=points.lines,pch=3,
       mkh=0.03,cex=0.4)
```

4.4.3 (b) matplot

- plottet die Spalten einer Matrix `x` gegen die einer anderen `y`; genauer:
  - `x[,1]` wird gegen `y[,1]`, `x[,2]` gegen `y[,2]` u.s.w. geplottet.

- Hat eine der beiden Matrizen weniger Spalten als die andere, werden die Spalten wie üblich zyklisch aufgefüllt;
- insbesondere: `x` oder `y` kann ein Vektor sein, gegen den dann alle Koordinaten von `y` resp. `x` geplottet werden;

- Syntax:

```
matplot(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL,
        col = 1:6, cex = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL,..., add = FALSE,
        verbose = getOption("verbose"))
```

- Argumente
  - wie bei **plot**
- entsprechend zu **points**, **lines**: **matpoints**, **matlines**

4.4.3 (c) weitere Methoden in R / S-Plus

- *Balloon–Plots*: balloonplot in Paket gplots
- *Star/Radar–Plots*: **stars** — Darstellung als Sterne/Spinnen
- *Symbol–Plots*: **symbols** — Darstellung als Kreise, Quadrate, Rechtecke, Sterne, Thermometer oder Boxplots
- *Chernoff–Gesichter*: faces — Darstellung als Gesichter; in R in Paket aplpack
- Übersicht über Graphik-Möglichkeiten in R: die R-Graph Gallery — http://addictedtor.free.fr/graphiques/

nur in S-Plus:

- 2D–*Histogramme*$^{S+}$: hist2d
- *Bi-Plots*$^{S+}$: biplots — gemeinsame Darstellung von Zeilen– und Spaltenraum einer multivariaten Variable in einem Plot

- *brush and spin*[S+]: brush — interaktives Betrachten der Daten durch Rotation und Markierung von Teilmengen
- nicht in R $\rightsquigarrow$ hier nur eine Demo, siehe auch session8.r

4.4.4 Interaktive Graphik

- ist eigentlich kein Ziel von R
- delegiert an andere Pakete $\rightsquigarrow$ Interfaces

4.4.4 (a) Zusatzpaket für interaktive Graphik: `TclTk`

- basiert auf `Tcl` (= *T*ool *C*ommand *L*anguage), siehe auch `http://www.msen.com/~clif/TclTutor.html` und dem dafür plattformübergreifend zur Verfügung stehenden Toolkit (`Tk`) für GUIs [*lies "tickeltikej"*]
- Interface von R zu `tcltk`
  - Autor ist Peter Dalgaard
  - wird bei der Standardinstallation automatisch mitinstalliert
- zum Anfangen

  - Artikel "A Primer on the R-Tcl/Tk package" und "Changes to the R-Tcl/Tk package" von Peter Dalgaard
  - Beispiele `tkdensity` und `tkttest` im Paket `tcltk` — abzurufen mit **demo**(tkdensity), **demo**(tkttest)
  - Beispiele von James Wettenhall auf
    http://www.sciviews.org/_rgui/tcltk/

- auf dieser Basis: die plattformunabhängige GUI (graphische Benutzeroberfläche) von J. Fox: R-Commander (Paket `Rcmdr`)

4.4.4 (b) Zusatzpaket für interaktive Graphik: `XGobi/GGobi`

- entwickelt bei ATT von Swayne, Cook, Buja
- zunächst für X–Server geschrieben — frei verfügbar unter
  http://www.research.att.com/areas/stat/xgobi/
- fortentwickelt auf Basis von `Gtk` zu `GGobi`
- ausführliches Manual unter http://www.ggobi.org/manual.pdf

- Interfaces gibt auch für R $\longleftrightarrow$ `xgobi` für `Windows` und `Unix`
- Interfaces von R zu `X/GGobi`
  - für `XGobi`: Paket xgobi
    - genauer steht dieses Interface in Version `1.2-5` unter dem angegebenen Link zur Verfügung
    - Autoren sind Martin Maechler und Kurt Hornik
  - für `GGobi`: Paket Rggobi
    - Autor ist Duncan Temple Lang
- Ziel: Visualisierung hochdimensionaler Daten
- Ideen:
  - Daten in mehreren Perspektiven simultan in verschiedenen Fenstern anschauen
    - Projektionen auf zwei Koordinaten
    - Verwendung "paralleler" Koordinaten
  - interaktiv

    - Beobachtungen identifizieren $\rightsquigarrow$ Labelling
    - Untergruppen in anderer Farbe einfärben (transient/permanent) $\rightsquigarrow$ Brushing
  - dabei stets entsprechende Reaktion der anderen Fenster — einfärben, labeln

- Demo mit

```
library(Rggobi)
?ggobi
ggobi(system.file("data", "flea.xml",
        package="Rggobi"), c("-noinit", "-xml"))
```


4.4.5 Filme

- Abschnitt beruht auf Hinweisen von Matthias Templ
- benötigt: entweder R-Paket `rgl+ImageMagick`, vgl. http://magick.imagemagick.org/script/index.php oder R-Paket `caTools`
- Vorgehen bei `rgl+ImageMagick`
  - Erzeugen einer Sequenz von png's mit rgl.snapshot
  - unter Windows: mit DOS-Kommando
    `convert -delay 10 *.png -loop 0 pic.gif`
    wird ein Film-`gif` erzeugt
  - unter Linux: komfortabler — Sequenz markieren und auf convert "werfen"
- Vorgehen bei `caTools`
  - Abspeichern der Folge von 2dim Graphiken als 3dim -Graphik mit der dritten Achse als Zeitachse

  - Erzeugen des `.gif`-Files mit write.gif

- Beispiel von Matthias Templ: `demorgl.R` und `mandelbrot2.R`

4.4.6 Flächendiagramme

- Für univariate Funktionen in zwei Variablen (z.B. $\mathbb{R}^2 \to \mathbb{R}$) kann man 3D–Plotfunktionen verwenden
- in R zur Verfügung stehende Methoden
  - perspektivischer 3D–Plot: **persp**
  - Plot der Niveaulinien: **contour**
  - farbkodierter Niveaulinienplot: **image**
- hier keine ausführliche Beschreibung der Funktionsweise
- siehe Demonstration und session8.r

4.5 Grafikaufbereitung

4.5.1 mehrere Plots in einem Diagramm

4.5.1 (a) mehrere Plots nebeneinander in einem/mehreren Fenster(n)

- erste Methode: (ein Fenster)
  - **par**(mfrow=c(<Zeilenzahl>,<Spaltenzahl>)),
  - **par**(mfcol=c(<Spaltenzahl>,<Zeilenzahl>))
- zweite Methode: (mehrere Fenster)
  - mit windows() oder **x11**() neues Fenster öffnen
  - **split** .**screen**(fig=c(<Zeilenzahl>,<Spaltenzahl>)),
  - interaktiv mit **split** .**screen**(prompt.screen()),
  - ungleiche Größen mit
    split .**screen**(fig=(matrix(<M.-el>,<Zahl M.-el>,4)),
    wobei jede Zeile der Matrix von der Form
    (<links>,<oben>,<unten>,<rechts>) ist und alle Koordinaten in $[0, 1]$ sind;

  - Zugriff mit **screen**(<Fig.Nr>),
  - Schließen einzelner / aller Figuren/Screens mit **close.screen**(<Fig.Nr>), bzw. (all=T)

- dritte Methode: (ein Fenster)
  layout — siehe auch Referat Matthias Brandl, Abschnitt 4.2
- Weiterschalten von verschiedenen Figuren: **frame**

4.5.1 (b) mehrere Plots übereinander in einer Figur

- um dafür zu sorgen, dass in dieselbe Figur geplottet wird:
  par(**new**=T) ab dem zweiten überlagerndem Plot
- um für eine einheitliche Skalierung zu sorgen: Verwendung von
  xlim, ylim
- um für eine einheitliche Beschriftung zu sorgen:
  - Setzen von xlab="", ylab="" in allen bis auf dem letzten Plot
  - Beschriftung durch xlab, ylab, **title**, **sub** erst im letzten Plot

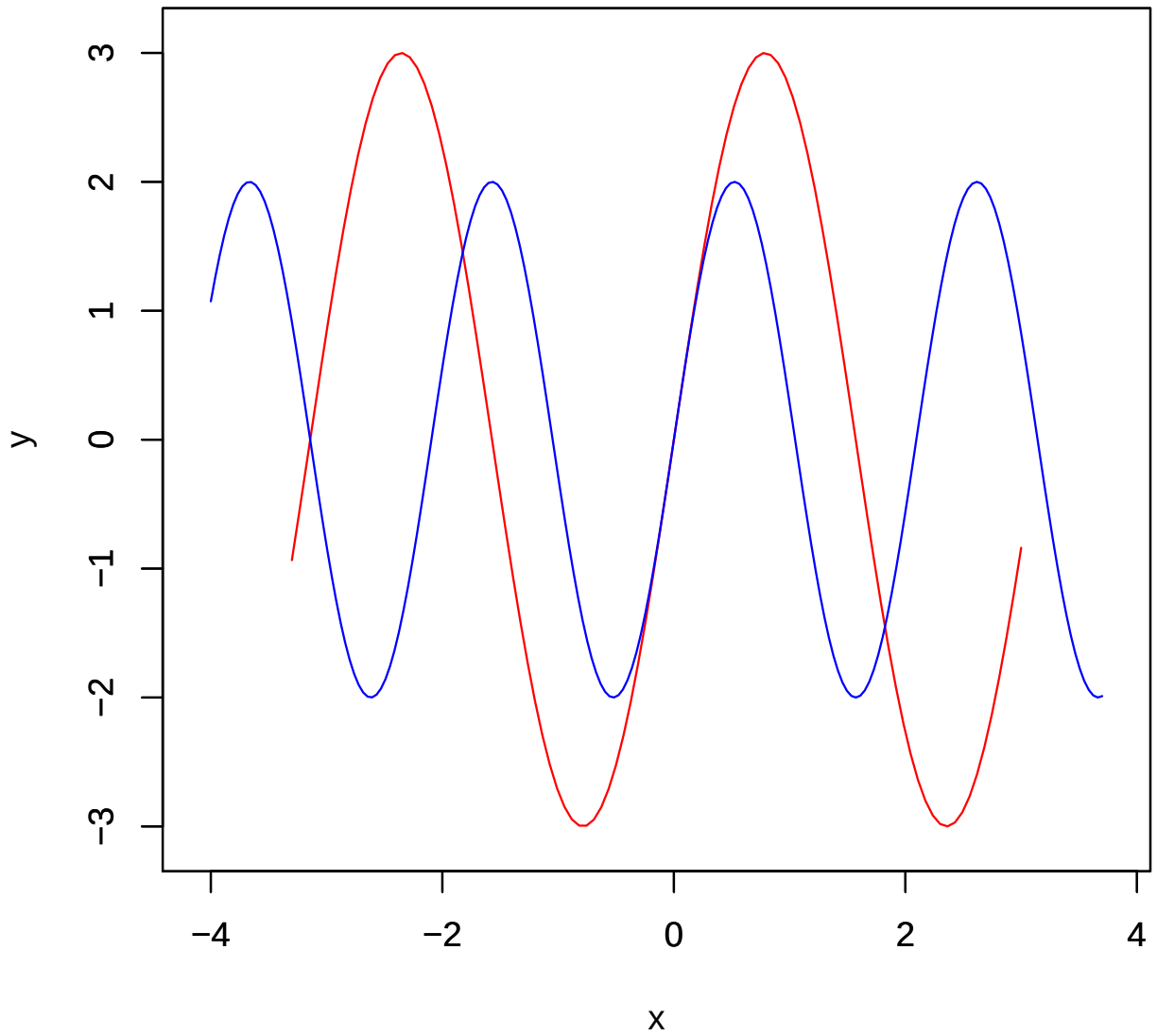
- neue Linien / Punkte in bestehenden Plot ohne weiteren Aufruf von **plot**(): line(), **points**(), matline(), **matpoints**() **abline**(),....;
-

R–Beispiel 4.5-1 [Überlagern mit untersch. Gittern]:

```
x1 ← seq(-3.3,3,length=100)
y1 ← 3*sin(2*x1)
x2 ← seq(-4,3.7,length=200)
y2 ← 2*sin(3*x2)
   #beachte: unterschiedliche Gitter!
xg ← c(min(c(x1,x2))-0.1,
       max(c(x1,x2))+0.1)
yg ← c(min(c(y1,y2))-0.1,
       max(c(y1,y2))+0.1)
plot(x1,y1,type="l",col="red",xlim=xg,
     ylim=yg,xlab="",ylab="")
par(new=T)
```


```
plot(x2,y2,type="l",col="blue",xlim=xg,
     ylim=yg,xlab="x",ylab="y")
title(paste("zwei verschiedene Funktionen",
      "f1,f2 in einem Plot"))
```


Peter Ruckdeschel
Matthias Kohl


zwei verschiedene Funktionen f1,f2 in einem Plot
y
x
3
2
1
0
−1
−2
−3
−4
−2
0
2
4

4.5.2 Hinzufügen von Information

4.5.2 (a) Einfügen von Linien, Punkten etc.

- Befehle: **abline**, **polygon**,**points**, **lines**, **segments**, **arrows**, **text**
- Beachte die Verwendung von `Expressions` wie in Abschnitt 4.5.4 zum Erzeugen mathematischer Formeln / Symbole
- siehe auch Referat Matthias Brandl, Abschnitt 4.2

4.5.2 (b) zusätzliche Achsen und Gitter

- manchmal sinnvoll: mehrere Achsen parallel — z.B. Celsius und Fahrenheit
- gelöst durch den Befehl **axis**

- Syntax:

  axis`(side, at = NULL, labels = TRUE, tick = TRUE, line = 0,`
  `pos = NA, outer = FALSE, font = NA, vfont = NULL, ...)`

- Argumente:
  - `side`: Zahl zwischen 1 und 4 — $\hat{=}$ `unten`, `links`, `oben`, `rechts`,
  - `at`: Vektor von Punkten, an denen Achsen–Ticks gezeichnet werden sollen
  - `labels`:
    - entweder (logisch) — numerische Beschriftung der Achsen–Ticks?
    - oder ein Stringvektor dessen Elemente an die Achsen–Ticks geschrieben werden sollen
  - `tick`: (logisch) sollen Achsen–Ticks gezeichnet werden?
  - `line`: Abstand mit der die Achse zur Originalachse gezogen wird; setzt den Wert des **par**–Parameters `mgp[3]` außer Kraft;

die relative Position der Achsen–Ticks bleibt unverändert

- `pos`: Position der (neuen) Achse in der anderen Koordinate; setzt die Werte von `line` und `mgp[3]` außer Kraft.
- `outer`: (logisch) Soll die Achse in den äußeren Rand statt in den "normalen" Rand geschrieben werden?
- `font`: Schrifttyp für den Text
- `vfont`: Vektor–Schrifttyp (`TT` für den Text
- `tck`: Länge der Achsen–Ticks (zwischen 0 und 1)
- `...`: weitere Graphikparameter

4.5.2 (c) Legenden

- mit der Funktion **legend** können Legenden eingefügt werden
- Syntax:

```
legend(x, y, legend, fill, col = "black", lty, lwd, pch,
       angle = NULL, density = NULL, bty = "o",
       bg = par("bg"), pt.bg = NA, cex = 1,
```


```
xjust = 0, yjust = 1, x.intersp = 1,
y.intersp = 1, adj = 0, text.width = NULL,
merge = do.lines && has.pch, trace = FALSE, ncol = 1,
horiz = FALSE)
```

- Argumente
  - `x`, `y`: x– und y–Koordinaten der Legende
  - `legend`: ein Stringvektor oder eine `Expression` wie in Abschnitt 4.5.4 der Länge ≥ 1 mit dem Inhalt der Legende
  - `fill`: falls spezifiziert werden Boxen mit der Farbe gefüllt
  - `col`, `lty`, `lwd`, `pch`, `cex`: Parameter für Linien, Punkte, Symbole in der Legende — wie in **plot**
  - `angle`, `density`: Schraffurparameter wie in **barplot**
  - `bg`: Hintergrundfarbe der Legenden–Box
  - `pt.bg`: Hintergrundfarbe der Punkte
  - `xjust`, `yjust`: wie die Legenden–Box ausgerichtet wird; $0\hat{=}$ links/unten $0.5\hat{=}$ zentriert und $1\hat{=}$ rechts/oben.

- `x.intersp`, `y.intersp`: Zeichenabständen (horizontal/vertikal)
- `adj`: die Stringausrichtung des Legendentextes; nützlich bei Verwendung von `Expressions` wie in Abschnitt 4.5.4
- `text.width`: Breite des Legendentextes in Benutzerkoordinaten;
- `merge`: (logisch) Sollen Punkte und Linien vereinigt werden?
- `trace`: (logisch) Sollen die Berechnungen von **legend** transparent sein?
- `ncol`: Zahl der Spalten, in die die Legende aufgeteilt werden soll
- `horiz`:(logisch) soll die Legende horizontal statt vertikal gesetzt werden? — setzt `ncol` außer Kraft.

- Rückgabewert: eine (unsichtbare) Liste von
  - `rect`: eine Liste mit Komponenten
    * `w`, `h` (positive Zahlen: die Breite und Höhe der

Legenden–Box)

  * `left`, `top` x– und y–Koordinaten der linken oberen Eckpunktes der Box

- `text`: eine Liste mit Komponenten: `x`,`y` — numerische Vektoren in der Länge der Legende, mit den x– und y–Koordinaten des Legendentexts

4.5.2 (d) Titel, Randbeschriftungen

- die Befehle **title** , **mtext**
- siehe Referat Matthias Brandl, Abschnitt 4.2
- man beachte auch die Verwendung von `Expressions` wie in Abschnitt 4.5.4 zum Erzeugen mathematischer Formeln / Symbole

4.5.3 interaktives Bearbeiten

- manchmal sinnvoll: Punkte interaktiv identifizieren, Objekte interaktiv platzieren
- Identifizieren von Punkten / Beobachtungen durch **identify**
- Syntax: **identify** `(x, ...)` oder

```
identify (x, y = NULL, labels = seq(along = x),
          pos = FALSE, n = length(x), plot = TRUE,
          offset = 0.5, ...)
```

- Argumente
  - `x`, `y`: Koordinaten der Punkte eines Scatterplots oder alternativ irgendein Objekt das Koordinaten definiert (z.B. eine Plot–Struktur eine Zeitreihe etc.); Übergabe auch als `x` allein möglich (`y` dann undefiniert).

- `labels`: ein optionaler Stringvektor der Länge von `(x,y)`, mit Namen/Labeln für die Punkte.
- `pos`: (logisch) — soll eine zusätzliche Komponente zum Rückgabewert hinzugefügt werden, der angibt, wohin der Name in Bezug auf das zu identifierende Objekt geplottet werden soll — ($1\hat{=}$ `unten`, $2\hat{=}$ `links`, $3\hat{=}$ `oben`, $4\hat{=}$ `rechts`).
- `n`: Maximalzahl an zu identifizierenden Objekten
- `plot`: (logisch) sollen die Label an die Punkte geschrieben werden?
- `offset`: der Abstand (in Zeichenweite) in dem die Label zum Objekt geschrieben werden;
- `...`: weitere Argumente an **par**.

- Details:
  - ist `plot TRUE`, so wird der identifizierte Punkt mit einem Label versehen;

  - dieser Label wird je nach Klickposition (in Bezug auf den identifizierten Punkt) unten, links, oben oder rechts gesetzt;
  - der Identifikations–Prozess wird durch Klicken mit der rechten Maustaste und anschliessender Auswahl des Menupunktes 'Stop' im Grafikfenster abgeschlossen
  - wird das Grafikfenster vor Abschluss des Identifikations–Prozesses umskaliert und anschließend das Fenster neu gezeichnet, verschwinden alle bis dahin identfizierten Label; sie werden erneut erscheinen, sobald der Identifikations–Prozess abgeschlossen ist und das Fenster neugezeichnet wird.

- Rückgabewert
  - ist `pos FALSE`, wird ein Integer–Vektor zurückgegeben mit den Indizes der identifizierten Punkte.
  - sonst wird dieser Vektor als Komponente `ind` mit einem weiteren Integer–Vektor — dieser als `pos` — zu einer Liste

zusammengefasst, in dem dann der Code für die relative Position des Labels steht

- **identify** wird im allgemeinen zusammen mit **locator** benutzt;
- **locator** liest die aktuelle Cursor–Position bei Klicken der linken Maustaste;
- Syntax: **locator**`(n = 512, type = "n", ...)`
- Argumente
  - `n`: wie bei **identify**
  - `type`: Punkt/Linientyp wie bei `plot`
  - `...`: weitere Graphikargumente, die genutzt werden, sofern `type` nicht `"n"` ist
- Beispiel: vergleiche 0.7

4.5.4 Mathematik in Labels

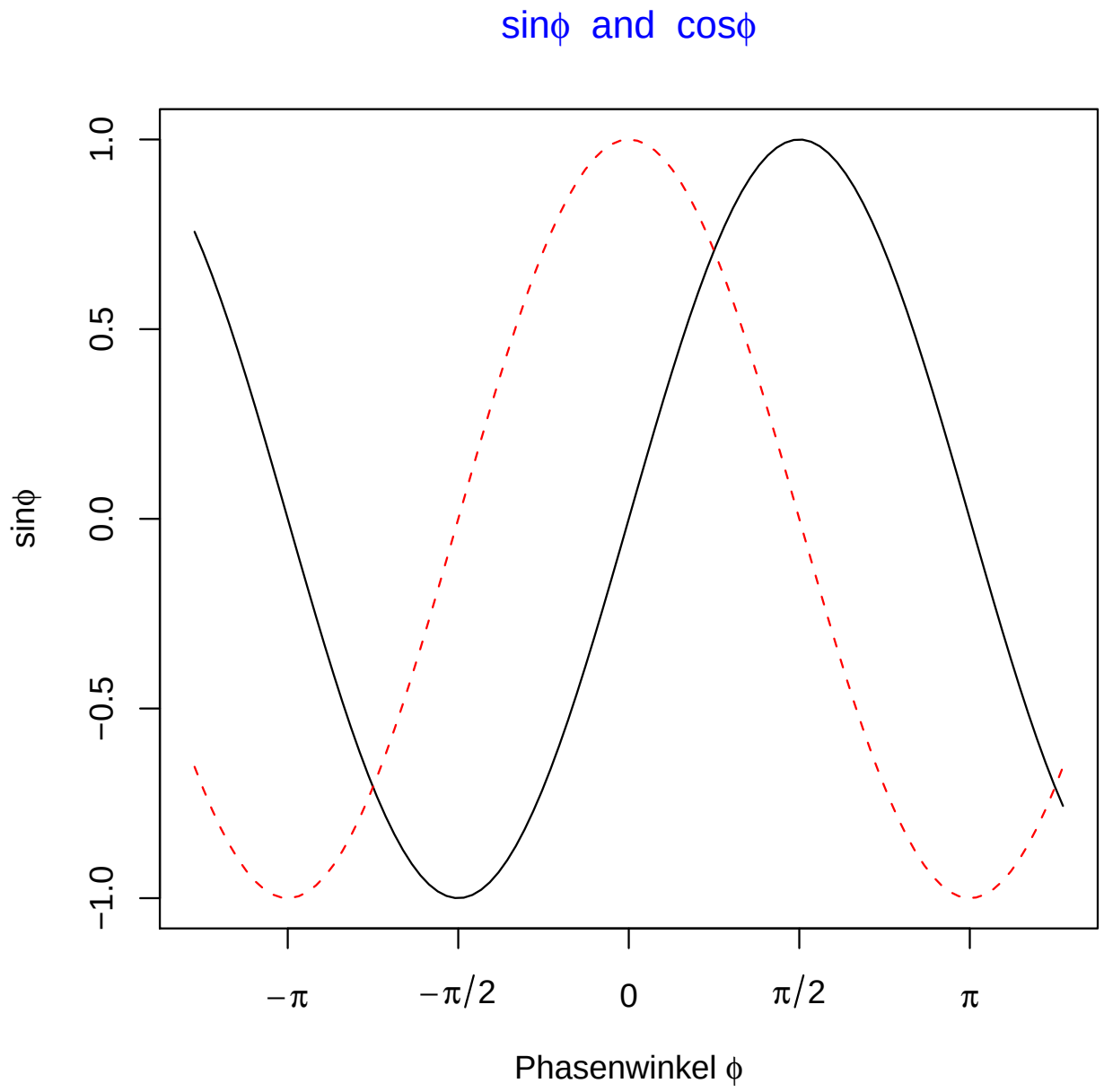
- vor allem im mathematischen Kontext, aber auch sonst, ist es oft nötig Formelzeichen wie Integrale, Brüche u.s.w. exakt darzustellen — sei es in Überschriften, in Legenden, in Achsenbeschriftungen
- in der mathematischen Fachliteratur hat sich hierzu TEX / LATEX durchgesetzt
- in R ist so etwas ähnliches möglich, indem der in "Mathematik–Modus" zu plottende Text als `expression` deklariert wird
-

R–Beispiel 4.5-2 [mathem. Ausdrücke in Überschrift]:

```
x ← seq(−4, 4, len = 101)
y ← cbind(sin(x), cos(x))
matplot(x, y, type= "l", xaxt= "n",
```


```
  main= expression(paste(plain(sin) * phi,
          "␣␣and␣␣", plain(cos) * phi)),
  ylab= expression("sin" * phi, "cos" * phi),
          # only 1st is taken
  xlab= expression(paste("Phasenwinkel␣", phi)),
  col.main= "blue")
axis(1, at= c(-pi, -pi/2, 0, pi/2, pi),
  lab= expression(-pi, -pi/2, 0, pi/2, pi))
```

weitere Beispiele siehe **help**(plotmath) bzw. in session8.r


sinϕ and cosϕ
sinϕ
1.0
0.5
0.0
−0.5
−1.0
−π
−π/2
0
π/2
π
Phasenwinkel ϕ

4.6 Bedingte Plots

- Situation: drei abhängige Variablen X, Y und Z
- Ziel: graphische Darstellung der Relation $X \leftrightarrow Y$ gegeben $Z = z$ oder $Z \in A_z$ für vorgebene Werte z oder Regionen A_z
- in R: durch **coplot** — Berechnung der Regionen durch **co**. intervals
- Spezifikation des "Ziels" durch Eingabe einer Formel — siehe auch Abschnitt 7.1.1 (b) — vom Typ X~Y|Z
- Literatur: Cleveland (1993)
- Syntax: **co**. intervals `(x, number = 6, overlap = 0.5)` und

```
coplot(formula, data, given.values, panel = points, rows,
       columns, show.given = TRUE, col = par("fg"),
       pch = par("pch"), bar.bg = c(num = gray(0.8),
       fac = gray(0.95)), subscripts = FALSE,
```


```
        xlab = c(x.name, paste("Given :", a.name)),
        ylab = c(y.name, paste("Given :", b.name)),
        axlabels = function(f) abbreviate(levels(f)),
        number = 6, overlap = 0.5, xlim, ylim, ...)
```

- **co**. intervals `(., number, .)` gibt eine $(\texttt{number} \times 2)$–Matrix `ci` zurück, wobei `ci[k,]` das k–te Bedingungsintervall an `x`–Werten ist
- Argumente
  - `x`: ein numerischer Vektor
  - `formula`: (Formel) die Formel die den Co-Plot definiert; zu deren Syntax siehe Abschnitt 7.1.1 (b) zwei Typen: `y ~ x | a` und `y ~ x| a * b`; bei letzterer wird nach den gemeinsamen Werten von `(a,b)` bedingt
  - `data`: ein Data–Frame, der alle Beobachtungen in den Variablen `(x,y,a[,b])` enthält
  - `panel`: (Funktion vom Typ **function**`(x, y, col, pch, ...)`):

gibt an, welche Aktion in jedem der Panels des Fensters durchgeführt werden soll; per default: **points**

- `rows`, `columns`: die Panels werden in einer Tabelle abgelegt; `rows` gibt die Zeilenanzahl der Tabelle an, `columns` die Spaltenzahl
- `given.values`: Bedingungswerte:
  - ein Objekt (oder eine Liste von zwei Objekten), die die Bedingungswerte für `a` (und `b`) festlegt;
  - liegt nur `a` vor:
    - gewöhnlich eine Matrix mit zwei Spalten, deren Zeilen die Bedingungs–Intervalle angeben
    - auch ein einzelner Werte–Vektor ist möglich
    - oder eine Menge von Faktor–Niveaus (sofern die bedingende Variable ein Faktor ist)
    - das Resultat von **co**. intervals kann direkt als Argument für `given.values` verwendet werden
- `show.given`: (logisch) (möglicherweise der Dimension 2×2)

sollen die bedingten Plots der entsprechenden bedingenden Variable ausgegeben werden?

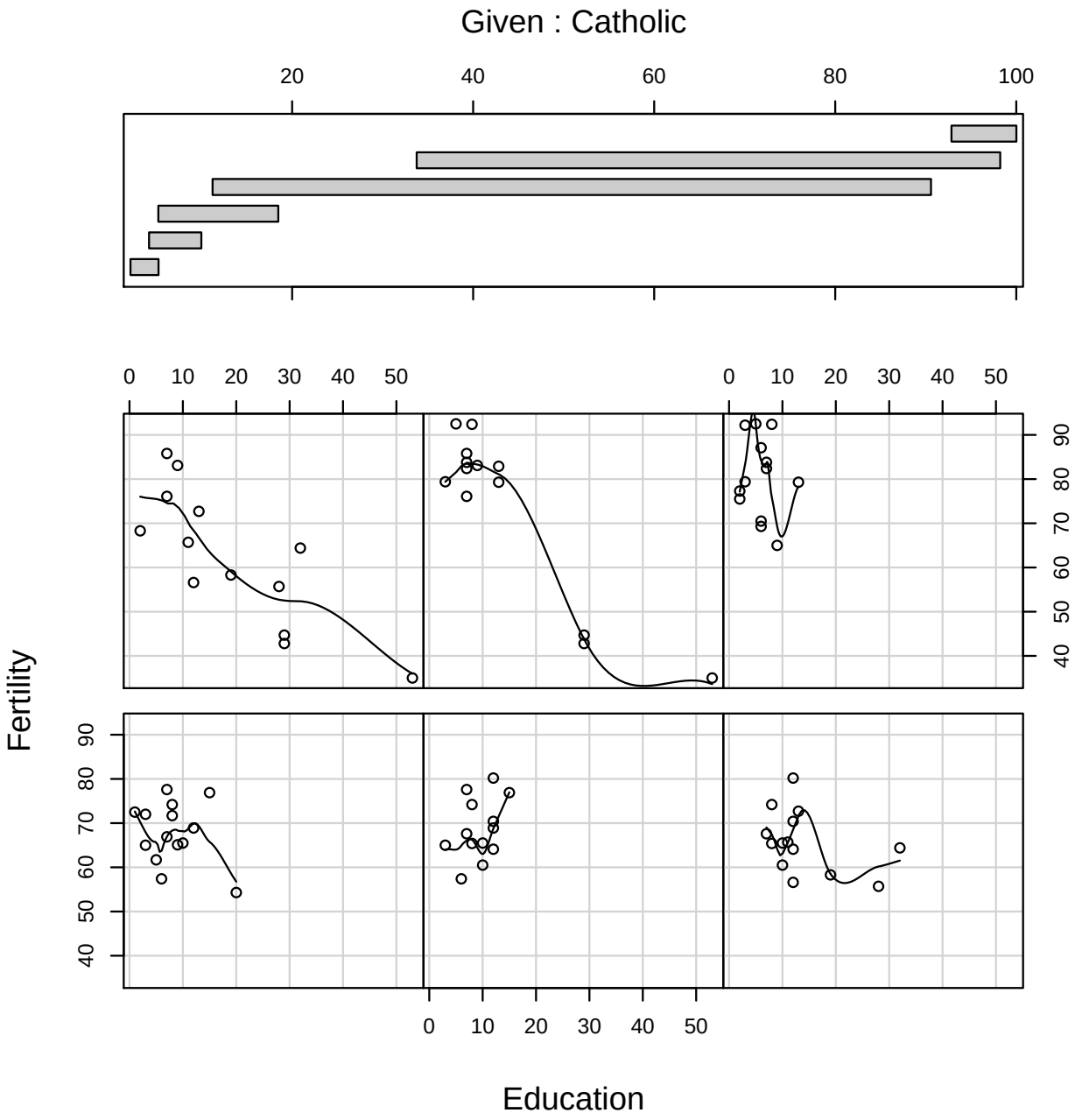
- `col`, `pch`: siehe **plot**
- `bar.bg`: ein Vektor mit Komponentennamen `"num"` und `"fac"` der die Hintergrundfarben der Säulen für die numerischen bzw. Faktor–wertigen Variablen angibt
- `xlab`, `ylab`: (String) Beschriftung für die x–Achse und die erste bedingende Variable (y–Achse und die zweite bedingende Variable). Ist nur ein Label angegeben, wird er für die x(y)–Achse verwendet und die default–Beschriftung für die bedingende Variable.
- `subscripts`: (logisch) Soll der Panel–Funktion ein zusätzliches (drittes) Argument `subscripts` übergeben werden, das die übergebenen Beobachtungen auf die in `subscripts` angegeben einschränkt?
- `axlabels`: (Funktion): erzeugt die

Achsen–Tick–Beschriftungen sofern `x` oder `y` Faktoren sind

- `number`: (Integer) Zahl der Bedingungsintervalle für `a`, `b` — möglicherweise der Länge 2. Wird nur benutzt, wenn die entsprechende bedingende Variable kein Faktor ist
- `overlap`: (numerisch), < 1; der Anteil an Überlappung der Bedingungsintervalle; möglicherweise der Länge 2 für x und y – Richtung. Ist `overlap < 0`, werden Lücken zwischen die Intervalle gelegt.
- `xlim`, `ylim`: der Plotbereich für die x(y)–Achse.
- `...`: weitere Argumente für die Panel–Funktion

R–Beispiel 4.6-1 [Bedingte Plots]:

```
#Einladen notwendiger Bibliotheken
library(MASS); library(modreg)
#Info ueber Datensatz
data(swiss); help(swiss)
#Wandeln in Data.frame
swiss.df<-data.frame(
        Fertility=swiss[,1],swiss[,-1])
#Schreiben einer Funktion zum simultanen Plotten
#der Punkte und einer lokalen robusten Regression
points.lines<-function(x,y,...)
    {points(x,y,...);
     lines(loess.smooth(x,y),...)}
#Erzeugen des Co-Plots
coplot(Fertility~Education | Catholic,
       data=swiss.df, panel=points.lines)
```

Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

4.7 Export von Daten und Graphik

nach einem Referat von *Matthias Kohl* vom 10.06.2002

Hinweis: Nicht (explizit) betrachtet werden hier die Exportmöglichkeiten via Verbindungen (*connections*) zu anderen Programmen wie etwa SPSS, SAS, SQL-Datenbanken, ...
(Diese Verbindungen sind auch in erster Linie für den Import von Daten vorgesehen)

4.7.1 Export von Daten

4.7.1 (a) Export–Funktionen

- `cat(..., file="", sep=" ", append=FALSE, ...)`:
  Schreibt die angegeben Objekte in ein File.
  `file=""`: Ausgabe erfolgt auf die Konsole.

- `write(x, file="data", ncolumns=if(is.character(x)) 1 else 5, append=FALSE)`:

  Schreibt die Daten aus `x` in ein file (default: ein File mit Namen `data`).

  `ncolumns`: Anzahl der Spalten, in die die Daten geschrieben werden

- `write.table(x, file="", append=FALSE, quote=TRUE, sep="", dec=".", row.names=TRUE, col.names=TRUE, ...)`:

  Schreibt die Daten aus dem Objekt `x`, in der Regel ein `Dataframe`, in eine Datei.

  `quote=TRUE`: es werden Anführungszeichen gesetzt

- `write.matrix(x, file="", sep=" ", blocksize)`:

  Bestandteil der `library` MASS
  Schreibt eine `Matrix` oder einen `Dataframe` in eine Datei
  Generiert eine "spaltenorientierte" Ausgabe

`blocksize >= 0`: das Object `x` wird in Blöcken mit jeweils `blocksize` Zeilen ausgegeben
(effiziente Speichernutzung, es können auch sehr große Matrizen ausgegeben werden, möglicherweise die Blöcke leicht unterschiedlich formatiert).

- `save(..., list=character(0), file="", envir=parent.frame(), ...)`

  Speichert die Objekte, die mit ihren Namen oder mit Hilfe eines `character` Vektors angegeben werden können.
  Zur Portierung von Objekten zwischen verschiedenen Plattformen/Rechnern
  `envir`: Frame, in dem nach dem Objekt gesucht wird.
  Zum Laden gespeicherter Objekte: `load()`

4.7.1 (b) Beispiel für den Export nach Excel

Vorgehensweise:

- Erzeuge einen `Dataframe` bzw. ein in einen `Dataframe` umwandelbares Objekt
- Exportiere in eine Datei:
  `write.table(X, file="a:export.txt", dec=",", row.names=FALSE)`
- Start von `Excel` und öffnen einer Textdatei

Hinweis: Es gibt auch die Möglichkeit `R` unter `Excel` aufzurufen und via einer `DCOM`-Verbindung zwischen `R` und `Excel` Daten auszutauschen (vgl. auch Abschnitt 1.3.4). Die notwendige Software findet man unter:
`http://cran.r-project.org/other-software.html`

4.7.2 Export von Graphik

4.7.2 (a) Graphik Devices

Mit Hilfe von `?Devices` Ausgabe der vorhanden Grafik Devices. Diese sind (unter Windows):

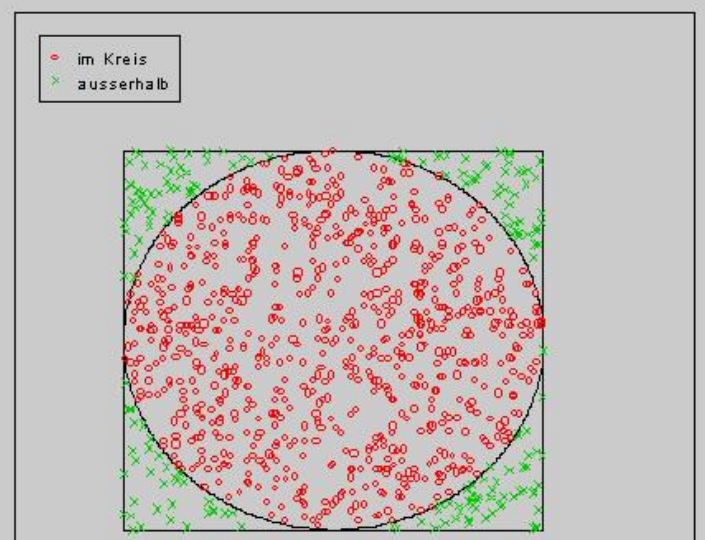
- `windows(...)`: Öffnet auf dem aktuellen Display ein neues Grafik Fenster
- `x11(...)`: Öffnet ein auf einem `X Window System` ein neues Grafik Fenster
- `postscript(...)`: vgl. Vorlesung
- `pdf(...)`: `pdf`-Dateien
- `pictex(...)`: `tex`-Dateien zur Verwendung in TEX bzw. LATEX
- `win.metafile(...)`: `wmf`-Dateien
- `win.print(...)`: Ausgabe auf Drucker

- `jpeg(...)`: jpeg-Dateien
- `png(...)`: png-Dateien, besser als `jpeg` für die Darstellung von Linien geeignet
- `bmp(...)`: bmp-Dateien
- `xfig(...)`: `fig`-Dateien zur Verwendung in `XFig` (version 3.2)
- `bitmap(...)`: verschiedene Grafik Formate (z.B. jpeg, png, tiff, ...), benötigt eine Version von `ghostscript`!
- `dev.off()`: Schließen der Grafik Device

4.7.2 (b) Beispiel für den Export nach PowerPoint

- Starten der Grafik Device: `win.metafile()`
- Erzeugen der Plots
- Schließen der Grafik Device: `dev.off()`

Prima Powerpointfolie



Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

5 Schätzen und Testen

5.1 klassische univariate Tests

5.1.1 Abriss Testtheorie

5.1.1 (a) Problemstellung

- Entscheidung zwischen Alternativen ist zu treffen
- zur Verfügung stehen Beobachtungen
- Voraussetzung: die Beobachtungen sind entweder gemäß (einem Element) der Hypothese H_0 (*Nullhypothese*) oder (einem Element) der Hypothese H_1 (*Alternative*) verteilt

5.1.1 (b) Typen von Tests

- Zahl d. Elemente in Hypothese: *einfach* $\leftrightarrow$ *zusammengesetzt*
- Art der Hypothese: *parametrisch* $\leftrightarrow$ *nichtparametrisch*

- "Richtung" der Hypothese: *einseitig* $\leftrightarrow$ *zweiseitig*
- bei diskreten Verteilungen: *randomisiert* $\leftrightarrow$ *nicht–randomisiert*
- nach Stichprobenumfang: *exakt* $\leftrightarrow$ *asymptotisch*
- nach Art der Beobachtungen: *Ein–* $\leftrightarrow$ *Zwei–Stichprobentest*

5.1.1 (c) Güte von Tests

- Fehlertypen: Fehler erster und zweiter Art (α– und β–Fehler)
- Verlustfunktion: Kombination —
  unter Schranke an den Fehler erster Art
  (*Signifikanzniveau*; ausgewertet über ganz H_0)
  Fehler zweiter Art minimieren
  (bzw. *Macht* maximieren; ausgewertet über ganz H_1)
- bei zwei– oder mehrseitigen Tests: *Unverfälschtheit*,
  i.e. Fehler zweiter Art nie über $1 - \alpha$
- α– und β–Fehler simultan durch größere Stichprobe verringerbar

5.1.1 (d) optimale Tests

- naheliegend: Finde diejenige Menge, auf der sich die unterstellten Maße gemäß H_0 bzw. H_1 maximal unterscheiden.
- $\rightsquigarrow$ interessant: Verteilung der Dichtequotienten (DQ)
  $\rightsquigarrow$ Neyman–Pearson–Lemma für *einfach* gegen *einfach*
- Test vom Neyman–Pearson–Typ:
  mit *Teststatistik/Prüfgröße* $T_n = T_n(X_1, \ldots, X_n)$ ergibt sich (ohne Randomisierung)

$$\varphi(X_1, \ldots, X_n) = \mathrm{I}_{\{T_n > c\}} =$$

$$= \begin{cases} 1 & \text{falls } T_n > c \quad \hat{=} \quad H_0 \text{ ablehnen} \\ 0 & \text{falls } T_n < c \quad \hat{=} \quad H_0 \text{ nicht ablehnen} \end{cases}$$

mit c dem *kritischen Wert* — natürlich je nach Kontext auch mit "$< c$"

- Ist $t^\natural$ die Realisation von T_n, so heißt die ZV $p_n := P(T_n > t^\natural)$ *p-value* und gibt das Signifikanzniveau an, zu dem der Test gerade noch ablehnen würde.
- bei monotonen DQ's: Übertragung der Optimalitätsaussage auf einseitige / unter Unverfälschtheit zweiseitige Tests
- i.a. nur asymptotische Aussagen möglich ($\rightsquigarrow$ ZGWS, Normalverteilung und abgeleitete Verteilungen)
- um Signifikanzniveau voll auszuschöpfen: bei diskreter Prüfgrößenverteilung Verbesserung der Macht durch *randomisierte Entscheidung*
- weitere Prinzipien (Stochastik II): Suffizienz, Invarianz

5.1.2 Gaußtest–Einstichprobenfall

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$, σ^2 bekannt, μ unbekannt
- Hypothesen: (analoges auch mit "$\geq$" $\leftrightarrow$ "$\leq$")
  - $H_0 : \mu = \mu_0 \quad$ vs $\quad H_1 : \mu = \mu_1 (< \mu_0)$
  - $H_0 : \mu \geq \mu_0 \quad$ vs $\quad H_1 : \mu < \mu_0$
  - $H_0 : \mu = \mu_0 \quad$ vs $\quad H_1 : \mu \neq \mu_0$
- Prüfgröße: $T_n = \bar{X}_n := \frac{1}{n} \sum_{i=1}^{n} X_i$, $\mathcal{L}_{H_0}(T_n) = \mathcal{N}(\mu_0, \sigma^2/n)$, (einseit. H_0) bzw. $T_n' = |\bar{X}_n|$ (zweiseit. H_0)
- Kritischer Wert: $[\mathcal{N}(\mu_1, \sigma^2/n)]^{-1}(\alpha)$
- Zweistichprobenfall analog — ebenfalls mit Gaußscher Prüfgrößenverteilung
- Umsetzung in R: so nicht implementiert;
  Ausweg: explizite Berechnung von c als

```
ccrit ←qnorm(alpha,mean=mu0,sd=sigma/sqrt(n))
```


5.1.3 t-Test für Mittelwert

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$,
  - Einstichprobenfall: σ^2 unbekannt, μ unbekannt,
  - Zweistichprobenfall: dazu sto. unabh. $Y_j \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu_0, \sigma^2)$, $j = 1, \ldots, m$, μ_0 unbekannt
- Hypothesen: (analoges auch mit "$\geq$"$\leftrightarrow$"$\leq$")
  - $H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu = \mu_1 (< \mu_0)$
  - $H_0 : \mu \geq \mu_0 \quad \text{vs} \quad H_1 : \mu < \mu_0$
  - $H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$
- Prüfgrößen:
  - Einstichprobenfall:
    - einseitiges H_0:
      $T_n = \frac{\sqrt{n}(\bar{X}_n - \mu_0)}{\sqrt{S_{n-1}}}$ mit $S_{n-1} := \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X}_n)^2$,
      $\mathcal{L}_{H_0}(T_n) = t_{n-1}$

    - zweiseitiges H_0: $T'_n = |T_n|$
  - Zweistichprobenfall:
    - einseitiges H_0:
      $T_{n,m} = \frac{\bar{X}_n - \bar{Y}_m}{\sqrt{(\frac{1}{n}+\frac{1}{m})(nS^X_{n-1}+mS^Y_{m-1})/(m+n-2)}}$
      mit S^X_n, S^Y_n den entspr. an X_i bzw. Y_j ausgewerteten Statistiken, $\mathcal{L}_{H_0}(T_{n,m}) = t_{m+n-2}$
    - zweiseitiges H_0: $T'_{n,m} = |T_{n,m}|$
- Kritischer Wert: $[t_{n-1}]^{-1}(\alpha)$ bzw. $[t_{n+m-2}]^{-1}(\alpha)$
- Umsetzung in R: als **t.test** bzw. pairwise.**t**.test

R–Beispiel 5.1-1 [T-test/Wilcoxon-test]:

```
x ← rnorm(10, mean=0, sd=2)
y ← rnorm(14, mean=1, sd=2)
t.test(x,y)
wilcox.test(x,y)
```


5.1.4 χ^2-Test für Varianzen

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$, σ^2 unbekannt, μ bekannt oder unbekannt
- Hypothesen: (analoges auch mit "$\geq$"$\leftrightarrow$"$\leq$")
  - $H_0 : \sigma = \sigma_0 \quad$ vs $\quad H_1 : \sigma = \sigma_1 (< \sigma_0)$
  - $H_0 : \sigma \geq \sigma_0 \quad$ vs $\quad H_1 : \sigma < \sigma_0$
  - $H_0 : \sigma = \sigma_0 \quad$ vs $\quad H_1 : \sigma \neq \sigma_0$
- Prüfgröße:
  - [μ bekannt] $T_n = n\tilde{S}_n/\sigma_0^2$ mit $\tilde{S}_n := \frac{1}{n}\sum_{i=1}^n (X_i - \mu)^2$, $\mathcal{L}_{H_0}(T_n) = \chi^2_n$
  - [μ unbekannt] $T_n = nS_{n-1}/\sigma_0^2$, $\mathcal{L}_{H_0}(T_n) = \chi^2_{n-1}$
- Kritischer Wert: $[\chi^2_{n[-1]}]^{-1}(\alpha)$, bzw. oberer und unterer Wert im zweiseitigen Fall;
- Umsetzung in R: so nicht implementiert; Ausweg: explizite Berechnung von c für $m = n[-1]$ als ccrit ←**qchisq**(alpha , **df**=m)

5.1.5 F-Test für Varianzen

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu_1, \sigma_1^2)$, $i = 1, \ldots, n$, dazu unabhängig, $Y_j \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu_2, \sigma_2^2)$, $j = 1, \ldots, m$, σ_k^2 unbekannt, μ_k bekannt oder unbekannt (hier unbekannt)
- Hypothesen: (analoges auch mit "$\geq$"$\leftrightarrow$"$\leq$")
  - $H_0 : \sigma_1 \geq \sigma_2$ vs $H_1 : \sigma_1 < \sigma_2$
  - $H_0 : \sigma_1 = \sigma_2$ vs $H_1 : \sigma_1 \neq \sigma_2$
- Prüfgröße: $T_{n,m} = \dfrac{S_{n-1}^X/(n-1)}{S_{m-1}^Y/(m-1)}$, $\mathcal{L}_{H_0}(T_{n,m}) = F_{n-1,m-1}$
- Kritischer Wert: $[F_{n-1,m-1}]^{-1}(\alpha)$, bzw. oberer und unterer Wert im zweiseitigen Fall;
- Umsetzung in R: als **var** . test

R–Beispiel 5.1-2 [F-test]:

```
x ← rnorm(10, mean=0, sd=2)
y ← rnorm(15, mean=0, sd=4)
var.test(x,y)
```


5.1.6 Binomialtest — Einstichprobenfall

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \text{Bern}(1, p)$, $i = 1, \ldots, n$, p unbekannt
- Hypothesen: (analoges auch mit "$\geq$"$\leftrightarrow$"$\leq$")
  - $H_0 : p \geq p_0 \qquad \text{vs} \qquad H_1 : p < p_0$
  - $H_0 : p = p_0 \qquad \text{vs} \qquad H_1 : p \neq p_0$
- Prüfgröße: $T_n = \#\{\text{Treffer}\}$, $\mathcal{L}_{H_0}(T_n) = \text{Bin}(n, p_0)$
- Kritischer Wert: $[\text{Bin}(n, p_0)]^{-1}(\alpha)$, bzw. oberer und unterer Wert im zweiseitigen Fall;
- Umsetzung in R: als binom.test
- asymptotisch auch in R: als **prop**.test

5.1.7 exakter Test von Fisher — Zweistichprobenfall

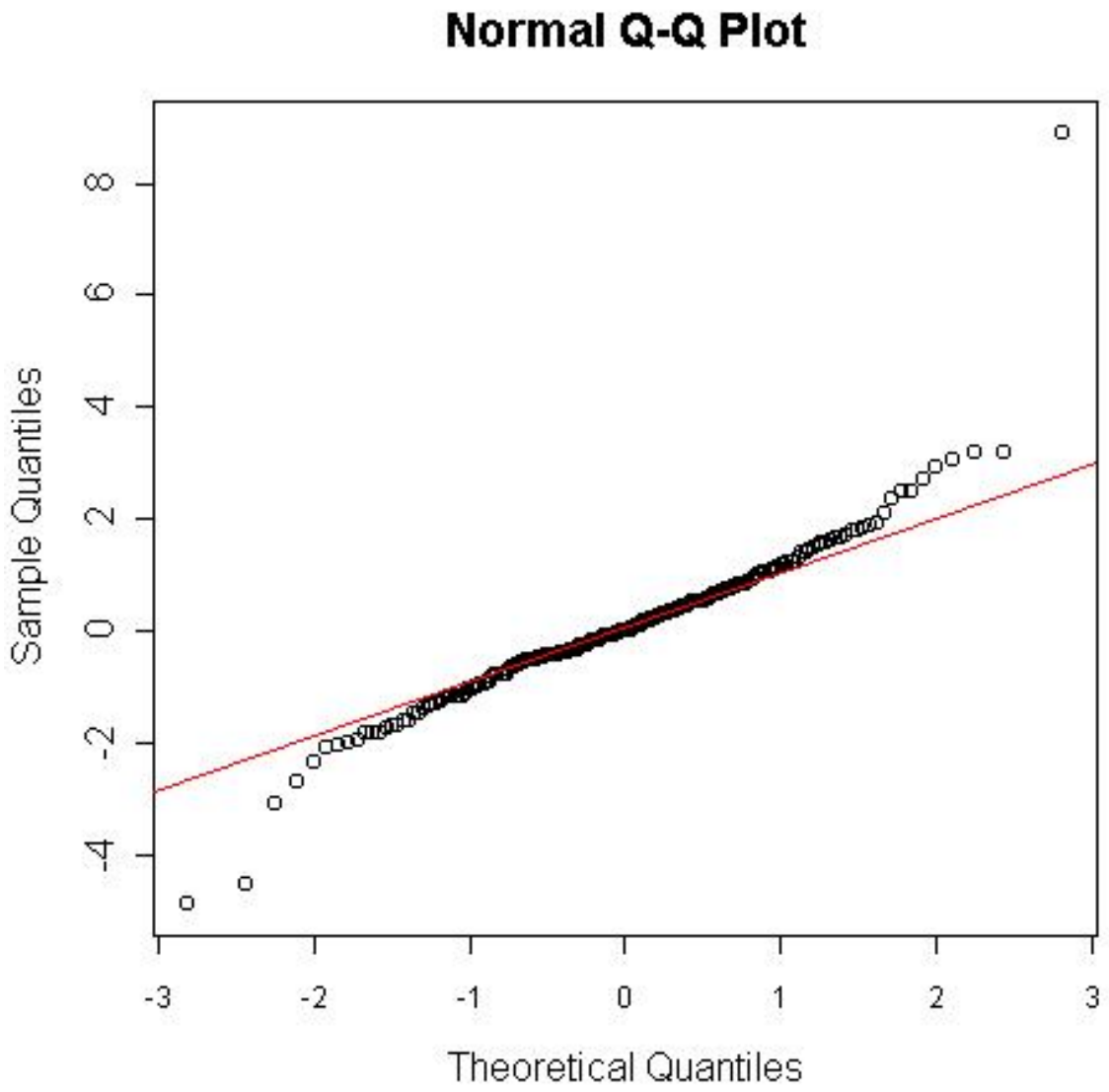
- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \text{Bern}(1, p_1)$, $i = 1, \ldots, n$, dazu unabhängig, $Y_j \overset{\text{u.i.v.}}{\sim} \text{Bern}(1, p_2)$, $j = 1, \ldots, m$, p_k unbekannt
- Hypothesen: (analoges auch mit "$\geq$"$\leftrightarrow$"$\leq$")
  - $H_0 : p_1 \geq p_2 \quad \text{vs} \quad H_1 : p_1 < p_2$
  - $H_0 : p_1 = p_2 \quad \text{vs} \quad H_1 : p_1 \neq p_2$
- Prüfgröße: $(U, V)_{n,m} = (\#\{\text{Treffer in } X_i\}, \#\{\text{Treffer in } Y_j\})^\tau$, $\mathcal{L}_{H_0}(U_{n,m} | V_{n,m} = v) = \text{HypGeo}(n + m, n, v)$
- Kritischer Wert: $[\text{HypGeo}(n + m, n, v)]^{-1}(\alpha)$, bzw. oberer und unterer Wert im zweiseitigen Fall;
- Umsetzung in R: (mit entsprechenden Identifikationen) als fisher . test
- asymptotisch auch in R: als **prop** . test

5.1.8 graphische Anpassungstests

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} P$, $i = 1, \ldots, n$
- Hypothese:
  $H_0 : P = P_0 = \mathcal{N}(\mu, \sigma^2)$ vs $H_1 : P \neq P_0 = \mathcal{N}(\mu, \sigma^2)$
  bzw.
  $$H_0 : P = P_0 \quad \text{vs} \quad H_1 : P \neq P_0$$
- graphische Überprüfung durch Plotten der Funktion
  $t \mapsto g(t) := F^{X;\text{emp}}(F_0^{-1}(t))$ für $t \in [0, 1]$
- gute Anpassung $\iff$ $g(t) \approx t$
- Umsetzung in R: **qqnorm** bzw. **qqplot**; Einzeichnen der Winkelhalbierenden durch **qqline**

R–Beispiel 5.1-3 [Graph. Test mit synthetischen Daten]:

```
# 200 Daten aus t5
y ← rt(200, df = 5)
# Vergleich mit Normalverteilung
qqnorm(y)
# zusaetzlich die Linie t|->t eingetragen
qqline(y, col = 2)
```



Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
-4
-2
0
2
4
6
8
-3
-2
-1
0
1
2
3

5.1.9 Shapiro–Wilk Normalverteilungstest

— siehe auch

http://www.itl.nist.gov/div898/handbook/prc/section2/prc213.htm

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} P$, $i = 1, \ldots, n$
- Hypothese: $H_0 : \; P = \mathcal{N}(\mu, \sigma^2)$ vs $H_1 : \; P \neq \mathcal{N}(\mu, \sigma^2)$
- Prüfgröße: $T_n = W = \frac{(\sum_i a_i X_{[i:n]})^2}{\sum_i (X_i - \bar{X}_n)^2}$, $\mathcal{L}_{H_0}(T_n)$ liegt tabelliert vor
- Kritischer Wert: liegt tabelliert vor;
- Umsetzung in R: als shapiro.test

5.1.10 Kolmogoroff(–Smirnoff)–Test

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} P$, $i = 1, \ldots, n$, P unbekannt, P_0 bekannt
  [oder dazu unabhängig, $Y_j \overset{\text{u.i.v.}}{\sim} P_0$, $j = 1, \ldots, m$
  und P_0 unbekannt],
- Hypothese: $H_0 : P = P_0 \qquad \text{vs} \qquad H_1 : P \neq P_0$
- (as.) Prüfgröße:
  - [P_0 bekannt]:
    $$T_n = \max_i \max\{|P_0(x_i) - \frac{i}{n}|,\ |P_0(x_i) - \frac{i-1}{n}|\},$$
    as. $- \mathcal{L}_{H_0}(T_n) = \text{Kolmogoroff}$, liegt tabelliert vor;
  - [P_0 unbekannt]:
    $$T_n = \frac{nm}{n+m} \sup_t |F^{X;\text{emp}}(t) - F^{Y;\text{emp}}(t)|,$$
    as. $- \mathcal{L}_{H_0}(T_n) = \text{Kolmogoroff}$
- Kritischer Wert: liegt tabelliert vor;
- Umsetzung in R: als ks.test

5.1.11 χ^2–Anpassungstest

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} \text{Multnom}(1, p_1, \ldots, p_{m-1})$, $i = 1, \ldots, n$, p_j unbekannt, Kategorien "1",...,"m"
- Bemerkung: entsteht auch durch Diskretisierung einer beliebigen Verteilungsfunktion F^X, indem wir die Realisationen von X klassieren
- Hypothese: $H_0 : p_j = p_j^{(0)} \; \forall j \qquad \text{vs} \qquad H_1 : p_j \neq p_j^{(0)} \; \exists j$
- Prüfgröße: $T_n = D_n^2 := \sum_{j=1}^{m} \frac{(\#\{\text{"}j\text{" in } \{X_i\}\} - np_j^{(0)})^2}{np_j^{(0)}}$,

  as.–$\mathcal{L}_{H_0}(T_n) = \chi^2_{m-1}$
- Kritischer Wert: $[\chi^2_{m-1}]^{-1}(\alpha)$
- Umsetzung in R: als chisq.test — Einteilung in Klassen in m Klassen mit **cut**

Peter Ruckdeschel
Matthias Kohl

R–Beispiel 5.1-4 [χ^2-Anpassungstest für Hypergeometrische-Variabl

```
## x ~ Hyper(m=10,n=13,k=9)
## nn =10 Wiederholungen
x <- rhyper(nn=10, m=10, n=13, k=9)
(x.class <- cut(x, -1:9))
(h.class <- table(x.class))
probs <- dhyper(0:9, m=10, n=13, k=9)
## sind die Daten HyperGeom-verteilt?
chisq.test(h.class, p = probs)
## kann nicht abgelehnt werden
probs <- dhyper(0:9, m=16, n=11, k=9)
chisq.test(h.class, p = probs)
## abgelehnt
```

R–Beispiel 5.1-5 [χ^2-Anpassungstest für Exp(2)-Variablen]

```
#X ~ Exp(lambda=2)
x <- rexp(50,rate=2)
## Zahl der Klassen ~ sqrt(n) : 7
p <- seq(0, 1, length = 8)

## lambda=2 bekannt:
x.class <- cut(x, qexp(p, rate = 2))
h.class <- table(x.class) ## Zellhaeufigkeiten
chisq.test(h.class)

## lambda=2 unbekannt:
## waehle lambda unguenstigst zur Ablehnung
## -> Statistik moeglichst klein
chi2dist <- function(lambda, XX){
    x.class <- cut(x, qexp(p, rate = lambda))
```


```
    h.class ← table(x.class)
    chisq.test(h.class)$statistic
}
chmin2dist ← function(XX, lambda = 1/mean(XX)){
    optimize(f = chi2dist,
             interval = c(1e-5,3*lambda),
             XX = XX)$minimum
}
(lb ← chmin2dist(x))
x.class ← cut(x, qexp(p, rate = lb))
h.class ← table(x.class)
chisq.test(h.class)
# => nicht ablehnen
```


5.1.12 Wilcoxon–Rangtest

- Modell: $X_i \overset{\text{u.i.v.}}{\sim} P$, $i = 1, \ldots, n$, P unbekannt,
  [oder dazu unabhängig, $Y_j \overset{\text{u.i.v.}}{\sim} P_0$, $j = 1, \ldots, m$, o.E. $m > n$,
  und P_0 unbekannt]
- Hypothesen:
  - H_0 : P ist sym. um 0 vs H_1 : P ist nicht sym. um 0
  - bei Paarung $(X, Y)_i$:
    H_0 : $\mathcal{L}(X - Y)$ ist symmetrisch um 0 vs
    H_1 : $\mathcal{L}(X - Y)$ ist nicht symmetrisch um 0
  - H_0 : P, P_0 haben selbes "Zentrum" vs
    H_1 : P, P_0 haben nicht selbes "Zentrum"
- Prüfgröße:
  - $T_n = \#\{\text{positive } X_i\} - \#\{\text{negative } X_i\}$, $\mathcal{L}_{H_0}(T_n)$ liegt tabelliert vor;

  - $T_n = \#\{\text{positive } (X_i - Y_i)\} - \#\{\text{negative } (X_i - Y_i)\}$, $\mathcal{L}_{H_0}(T_n)$ liegt tabelliert vor;
  - $T_{n,m} = \sum_i \mathrm{Rang}(X_i; \text{in } \{X_k\} \cap \{Y_j\})$, $\mathcal{L}_{H_0}(T_{n,m})$ liegt tabelliert vor;
- Kritischer Wert: liegt tabelliert vor;
- Umsetzung in R: als wilcox . test

5.1.13 Korrelationstest

- Modell: $(X_i, Y_i) \overset{\text{u.i.v.}}{\sim} P$, $i = 1, \ldots, n$,
- Hypothese $H_0 : \mathrm{R}(X, Y) = 0 \quad \text{vs} \quad H_1 : \mathrm{R}(X, Y) \neq 0$ mit R einem geeignetem Korrelationskoeffizienten, z.B.
  - klassischer Korrelationskoeffizient ρ:

$$\rho_{X,Y} = \frac{\mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]}{\sqrt{\mathrm{Var}[X]\,\mathrm{Var}[Y]}},$$

$$\rho_{X,Y}^{\text{emp}} = \frac{\sum_i (X_i - \bar{X}_n)(Y_i - \bar{Y}_n)}{\sqrt{(\sum_i (X_i - \bar{X}_n)^2)(\sum_i (Y_i - \bar{Y}_n)^2)}}$$

– Kendall's τ:

$$\tau_{X,Y} = 2P((X_1 - X_2)(Y_1 - Y_2) > 0) - 1 :$$

Sei $t_{i,j} := \text{sign}(X_i - X_j)\,\text{sign}(Y_i - Y_j)$. Dann ist

$$\tau_{X,Y}^{\text{emp}} := \sum_{i<j} t_{i,j} / \binom{n}{2}$$

– Spearman's r:

mit $r_i = \text{Rang}(X_i;\ \text{in}\{X_k\})$, $s_j = \text{Rang}(Y_j;\ \text{in}\{Y_k\})$ gilt:

$$\text{r}_{X,Y}^{\text{emp}} = \rho_{r_i,s_j}^{\text{emp}} :$$

- Prüfgröße: $T_n = \text{R}_n^{\text{emp}}$, $\mathcal{L}_{H_0}(T_n)$ liegt tabelliert vor;
- Kritischer Wert: liegt tabelliert vor;
- Umsetzung in R: als **cor** . test

5.2 Schätzen eines Parameters

5.2.1 Abriss der klassischen Schätztheorie

5.2.1 (a) Problemstellung

- parametrisches Modell:
  eine Familie $\mathcal{P}$ von W-Maßen P_θ auf $(\mathbb{R}, \mathbb{B})$ bzw. $(\mathbb{R}^k, \mathbb{B}^k)$
  parametrisiert durch Parameter θ aus einer offenen Menge
  $\Theta \subset \mathbb{R}^p$: $\mathcal{P} := \{P_\theta \,|\, \theta \in \Theta\}$
- $\rightsquigarrow \mathrm{E}_\theta$, Var_θ
- zur Verfügung stehen Beobachtungen $X_1, \ldots, X_n \overset{\text{u.i.v.}}{\sim} P_\theta$
- Aufgabe: Schätzung von $\tau(\theta)$, einer Funktion $\tau : \Theta \to \mathbb{R}^d$ durch messbare Funktionen $T_n(X_1, \ldots, X_n)$

- Angabe einer möglichst "genauen" (zufälligen!) Näherung für $\tau(\theta) \rightsquigarrow$ *Punktschätzung*
- Angabe einer möglichst "kleinen" (zufälligen!) Intervalls/Bereichs der $\tau(\theta)$ mit gegebener WS einfängt $\rightsquigarrow$ *Intervall–/Bereichsschätzung* resp. Konfidenzintervalle
- *Bayes–Schätzung*
  - Bewertung der Wahrscheinlichkeit der verschiedenen θ's vor Kenntnis der $X_1, \ldots, X_n \rightsquigarrow$ *à priori Verteilung*,
  - Auffassen von P_θ als bedingte Wahrscheinlichkeit gegeben θ,
  - mit $X_1, \ldots, X_n$ erstellen der *à posteriori Verteilung* für θ gegeben $X_1, \ldots, X_n$
  - Schätzung von θ durch Erwartungswert dieser *à posteriori Verteilung*

5.2.1 (c) Schätz–Methoden für Punktschätzer

- Momentenmethode
- Maximum–Likelihood–Prinzip
- M–Schätzer (Nullstellen einer M–Gleichung)
- L–Schätzer (Linearkombinationen von Ordnungsstatistiken)
- R–Schätzer (Schätzer basierend auf Rängen)
- Schätzer durch Bootstrap

5.2.1 (d) Schätz–Methoden für Bereichsschätzer

- exakte Konfidenzintervalle mit Quantilsfunktion
- approximative Konfidenzintervalle mit Tschebyscheff–Ungleichung
- approximative Konfidenzintervalle durch Normalapproximation

5.2.1 (e) Güte von Schätzern

- *Konsistenz*: Gilt $T_n \to \tau(\theta)$ stochastisch / fast sicher?
- *Unverzerrtheit*: Gilt $\mathrm{E}_\theta[T_n] = \tau(\theta)$ für alle θ?
- Bemessung der Abweichung von T_n zu $\tau(\theta)$ durch *Verlustfunktion*, z.B. $\|\cdot\|^2$
- Gütekriterium: *Risiko*, i.e. erwarteter Verlust, z.B. MSE, ist zu minimieren $\rightsquigarrow$ *Effizienz*
- Konzentration der vorhandenden Information $\rightsquigarrow$ *Suffizienz*
- Wahrung von Invarianzeigenschaften des Modells $\rightsquigarrow$ *Äquivarianz*

5.2.1 (f) optimale Schätzer

- Cramér–Rao–Schranke für die Varianz
- oft: Maximum Likelihood Schätzer (MLE) erreicht Schranke (zumindest asymptotisch)
- bei Beschränkung auf unverzerrte Schätzer: Prinzip "UMVU"
- bei Beschränkung auf lineare, unverzerrte Schätzer: Prinzip "BLUE"

5.2.2 Schätzen eines Parameters in R

5.2.2 (a) "direkte Programmierung"

- Situation: MLE explizit analytisch zu ermitteln und auch Verteilung ist bekannt
- dann: einfaches Verwenden der Routinen in R, z.B. `mean`, `IQR`, bzw. zur Genauigkeit `pbinom`, `qnorm`, ...
- Beispiel — vgl. auch Blatt 8, Aufgabe A.8.1
  - $X_1, \ldots, X_n \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, \sigma^2)$, σ bekannt, gesucht μ
  - Lösung: $\bar{X}_n$, also `mean`
  - Angabe von $1-\alpha$–Konfidenzintervallen durch $\bar{X}_n \pm \sigma/\sqrt{n}\,$ `qnorm`$(1-\alpha/2)$
- Beispiel in R — Schätzung von μ in $\mathcal{N}(\mu, \sigma^2)$, falls σ unbekannt mit $S_n^2 := \frac{1}{n}\sum_i (X_i - \bar{X}_n)^2$ und $\tilde{S}_n^2 := \frac{n}{n-1} S_n^2$ gilt

$$\sqrt{n}(\bar{X}_n - \mu)/\tilde{S}_n \sim t_{n-1}$$

R–Beispiel 5.2-1 ["direkte Programmierung"]:

```
X←rnorm(20)
# ML-Schaetzer: mean(X), var(X)
m←mean(X); V←var(X)
# Genauigkeit?
n←length(X)
sn1←(n/(n-1)*V)^0.5
t1←qt(0.975,df=n-1)/sqrt(n)*sn1
cat("Mittelwert der ersten Stichprobe wird mit",
    "95% Ws eingefangen durch [",
          m1-t1,",",m1+t1,"]\n")
# mit Bootstrap
Xb←numeric(1000)
for(i in 1:1000)
  Xb[i]←mean(sample(X,replace=T))
Xbo←sort(Xb)
```


```
cat("Mittelwert der ersten Stichprobe wird mit",
    "95% Ws eingefangen durch [",
        Xbo[25],",",Xbo[975],"]\n")
```

5.2.2 (b) numerische Optimierung

- Angabe der Likelihood als Funktion in $\Theta; X_1, \ldots, X_n$ möglich,
- aber:
  - Likelihood nicht glatt in θ
  - keine analytische Angabe des (globalen) Maximums möglich

  $\rightsquigarrow$ Auswertung auf Gitter und Nutzung von `which.max`
  - Beispiel: Blatt 8, Aufgabe A.8.2, sowie:
    Schätzung der Freiheitsgrade einer χ^2-Verteilung durch ML

    R–Beispiel 5.2-2 [Gittersuche]:

```
y ← rchisq(20, df=4)
#
```


```
dftry←1:40 # Gitterpunkte
likelihood←numeric(1:40)
for(i in 1:40)
     likelihood[i]←prod(dchisq(y,df=dftry[i]))
dfe←dftry[which.max(likelihood)]

#Genauigkeit mit Bootstrap
dfeb←numeric(1000)
for(i in 1:1000)
  {yb←sample(y,replace=T)
   for(j in 1:40)
       likelihood[j]←prod(dchisq(yb,df=dftry[j]))
   dfeb[i]←dftry[which.max(likelihood)]}
dfebo←sort(dfeb)
cat("wahre Zahl der Freiheheitsgrade wird mit",
    "95% Ws eingefangen durch [",
     dfebo[25],",",dfebo[975],"]\n")
```


- oder aber:
  - Parameter ist mehrdimensional
  - $\rightsquigarrow$ Nutzung der R–Funktionen `optim` oder `optimize`

5.2.2 (c) numerisches Lösen von Gleichungen

- wie im vorigen Abschnitt, nur Likelihood glatt in θ und $k = 1$
- Finden des Maximums mit differentiellen Methoden als Nullstelle
- allgemeiner: bei Verwendung von M–Schätzern stößt man meistens auf nicht explizit lösbare Nullstellenprobleme
- $\rightsquigarrow$ Lösung von $\sum_{i=1}^{n} \Lambda_\theta(X_i) = 0$ bzw. $\sum_{i=1}^{n} \psi_\theta(X_i) = 0$ in θ mit R–Funktion `uniroot`
- Beispiel: Blatt 8, Aufgabe A.8.3 (a), sowie: numerisches Lösen der ML-Gleichung im Fall

$$X_i - \theta \overset{\text{u.i.v.}}{\sim} \text{Cauchy}$$

Es gilt $\Lambda_\theta(x) = \Lambda_0(x - \theta)$ und $\Lambda_0(x) = 2x/(x^2 + 1)$

R–Beispiel 5.2-3 [Maximierung mit uniroot]:

```
x←c(2.942, 2.764, 2.007, 1.192, −4.989, 1.812,
    2.894, 1.179, 2.559, 1.478, −54.0169,
    1.382)
Lambda←function(theta,y)
    {sum((y−theta)/(1+(y−theta)^2))}
help(uniroot)
uniroot(Lambda,low=−20,up=20,
        tol=1e−10,y=x,maxiter=20)
#Zugriff auf NS mit $root
uniroot(Lambda,low=−20,up=20,tol=1e−10,
        y=x,maxiter=20)$root
#Genauigkeit mit Bootstrap
thb←numeric(1000)
for(i in 1:1000)
  {xb←sample(x,replace=T)
```


```
    thb[i] <- uniroot(Lambda, low=-20,
                      up=20, tol=1e-10, y=xb,
                      maxiter=20)$root}
  thbo <- sort(thb)
  cat("wahrer Parameter theta wird mit",
      "95% Ws eingefangen durch [",
          thbo[25],",",thbo[975],"]\n")
```

5.2.2 (d) Schätzgenauigkeit / Konfidenzintervall

- direkte Bestimmung (wie in Abschnitt 5.1.2.(a))
- theoretische Bestimmung mit Normalapproximation
- Bestimmung durch Bootstrap

5.2.3 Robuste Parameterschätzung in R

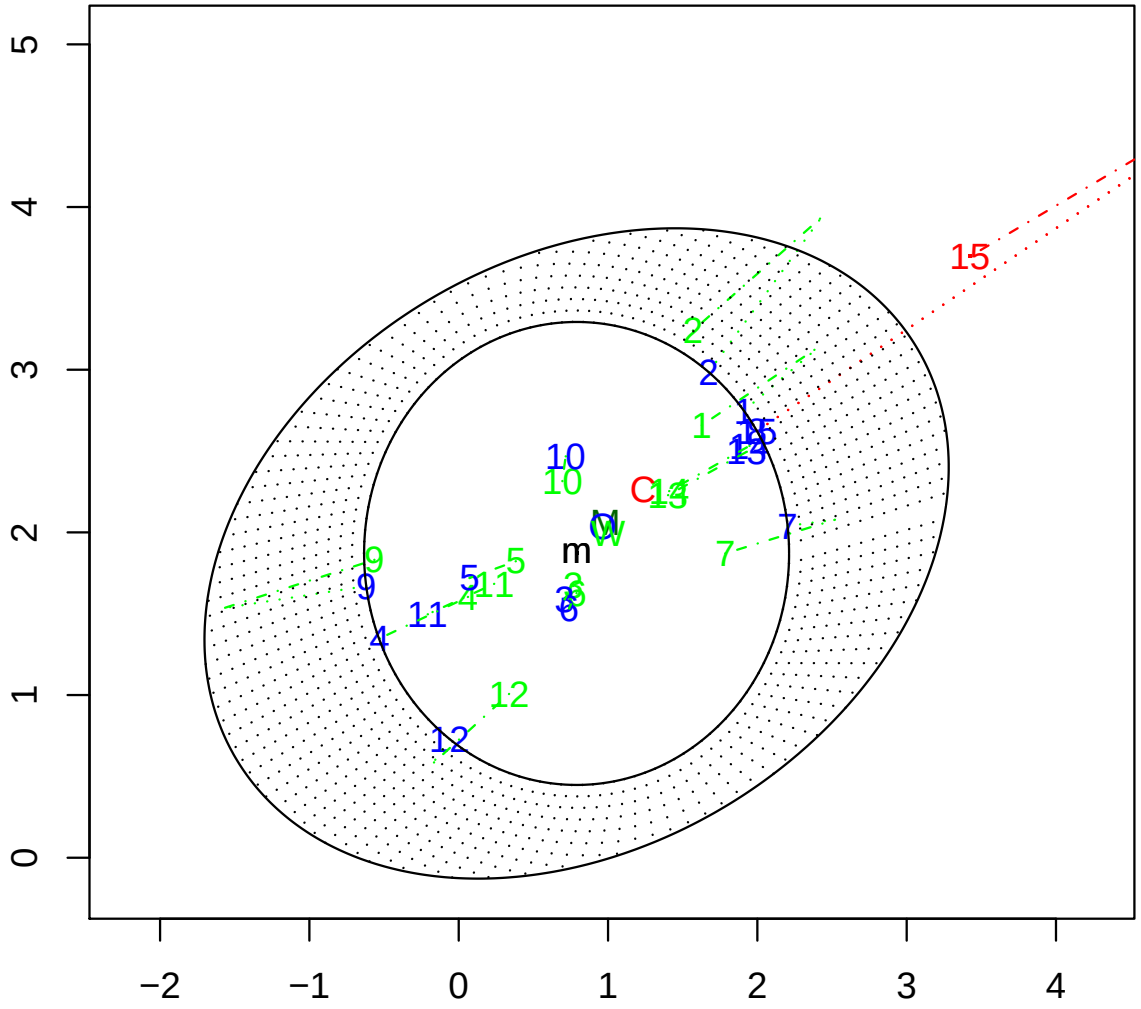
5.2.3 (a) ein einführendes Beispiel

Schätzung der Lokation unter Ausreißern

- Situation: p–dimensionale Beobachtungen $X_1, \ldots, X_n$
- (ideales) Modell: $X_i = \theta + u_i$, $\theta \in \mathbb{R}^p$ unbekannt, $u_i \overset{\text{u.i.v.}}{\sim} F$
- Kontaminations–Situation: $u_i \overset{\text{u.i.v.}}{\sim} (1-r)F + rH$,
  H unbekanntes und unkontrollierbares Maß
- konkret: $n = 36$, $p = 2$, $F = \mathcal{N}_2(0, \Sigma)$, $\Sigma = \begin{pmatrix} 0.87, & 0.29 \\ 0.29, & 0.38 \end{pmatrix}$,
  bekannt, $\theta = (1, 2)^\tau$ unbekannt
- klassisch optimales Verfahren: $\hat{\theta} = \bar{X}_n$
- unter Ausreißern: unbeschränkte Auslenkung!
- typische Anwender–Strategie: Verwerfe alle Beobachtungen X_i
  mit $|X_i|$ "groß" (*Hard Rejection*)

- Problem: Auch in den verworfenen Beobachtungen steckt Information über θ!
- Alternative: $\check{\theta} := (\mathrm{Med}X_{i;1}, \ldots, \mathrm{Med}X_{i;p})^\tau$; verwertet alle Beobachtungen, viel stabiler als $\bar{X}_n$, aber auch viel mehr Beobachtungen nötig, um gleiche Effizienz zu erhalten
- (asympt.) Optimierungsproblem: finde ein Verfahren, das für $n \to \infty$ den mittleren quadratischen Fehler (MSE), also $|\mathrm{Bias}|^2 + \mathrm{tr}\,\mathrm{Var}$ für alle Situationen, auch die kontaminierten, minimiert
- Lösung hier: *One–Step–Konstruktion*
  $\theta^{(0)} := \check{\theta} + \frac{1}{n}\sum_{i=1}^n A(X_i - \check{\theta})\min(1, \frac{b}{|A(X_i - \check{\theta})|})$
  dabei werden A und b aus impliziten Gleichungen berechnet (vgl. Übung)
- Interpretation als Pseudobeobachtungen:
  $Y_i := \check{\theta} + A(X_i - \check{\theta})\min(1, \frac{b}{|A(X_i - \check{\theta})|})$

- Interpretation der folgenden Grafik:
  - $W \hat{=} \theta$, $C \hat{=} \hat{X}_n$, $m \hat{=} \check{\theta}$, $M \hat{=}$ M–Schätzer, $O \hat{=} \theta^{(0)}$
  - Ziffern: Beobachtungsindex
  - in grün: ideale Original–Beobachtungen,
  - in rot: kontaminierende Original–Beobachtungen,
  - in blau: Pseudobeobachtungen,
  - Kreise/Ellipsen: der äußere ist das Bild des inneren unter A; Punkte auf dem inneren Kreis werden längs gestrichelter Linien abgebildet
  - — falls $|Ax| < b$ nur ein "Transport", sonst zuerst "Transport" mit A dann Projektion auf Kreis mit Radius b um m



Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

- qualitative Robustheit
  - Sind die Verfahren (gleichmäßig) stetig in der schwachen Topologie?
    d.h. konvergiert mein Schätzer stochastisch (ist konsistent) nicht nur für eine ideale Verteilung, sondern für alle Verteilungen, die in der "Nähe" dieser idealen Verteilung liegen?
  - dabei ist "nah" genau in dem Sinn zu verstehen, wie man sagt, dass im Zentralen Grenzwertsatz für $\mathrm{u.i.v.}$ – Beobachtungen die Verteilung von $\frac{1}{\sqrt{n}} \sum_i (X_i - \mathrm{E}[X_1])$ immer näher an $\mathcal{N}(0, \mathrm{Var}(X_1))$ liegt.
  - $\bar{X}_n$ ist in diesem Sinn unstetig, der Median stetig, sofern er eindeutig ist

- Ausreißer, Kontamination, Umgebungen um ideales Modell

  Nachbarschaften um ein ideales Modell:

  - Abstand P^{real} zu P^{ideal} nicht größer als r

    Abstandsbegriffe (Auswahl):

    * Kolmogoroffabstand $d_K(P,Q) := \sup_t |F^P(t) - F^Q(t)|$,
    * Totalvariationsabstand $d_V(P,Q) := \int |p-q|\, d\mu$,
    * Hellingerabstand $d_h(P,Q)^2 := \frac{1}{2}\int |p-q|^2\, d\mu$

  - Konvexkontaminationsumgbg / *Gross Error Model*:

    zu $r \in (0,1)$ werden alle Maße $\hat{P}$ von der Form

    $$\hat{P} := (1-r)P + r\tilde{P}$$

    simultan betrachtet, $\tilde{P}$ ein kontaminierendes Maß

  - Interpretation: mit Wahrscheinlichkeit r wird die Zufallsvariable mit Maß P ersetzt durch einen Ausreißer mit Verteilung/Maß $\tilde{P}$

  - Radius r steuert Grad der Robustheit

  - feststehende $\leftrightarrow$ schrumpfende Umgebungen

- Bruchpunkt:
  - bei wieviel Prozent Kontamination kann der Schätzer zusammenbrechen? anschaulich: wieviele Beobachtungen können beliebig manipuliert werden, ohne dass der Schätzer gegen $\pm\infty$ (bzw. Rand des Wertebereichs) getrieben werden kann
  - z.B. $\bar{X}_n$ hat Bruchpunkt 0, Median Bruchpunkt $1/2$

- Influenzkurve
  - Ableitungsbegriff für Schätzer / Funktionale
  - als Differentialquotient für ein Funktional
    $\mathrm{IC}(T,x) = \lim_{t\downarrow 0}(T((1-t)P + t\,\mathrm{I}_x) - T(P))/t$
  - als mögliche Linearisierung:
    ein Schätzer S_n (in einem glatten Modell) heißt *asymptotisch linear* in θ, falls es ein $\psi_\theta \in L_2^k(P_\theta)$ mit (5.2.1) und (5.2.2) gibt, so dass $S_n = \theta + \frac{1}{n}\sum_{i=1}^n \psi_\theta(x_i) + \mathrm{o}_{P_\theta}(\frac{1}{\sqrt{n}})$
  - Eigenschaften (5.2.1) und (5.2.2):

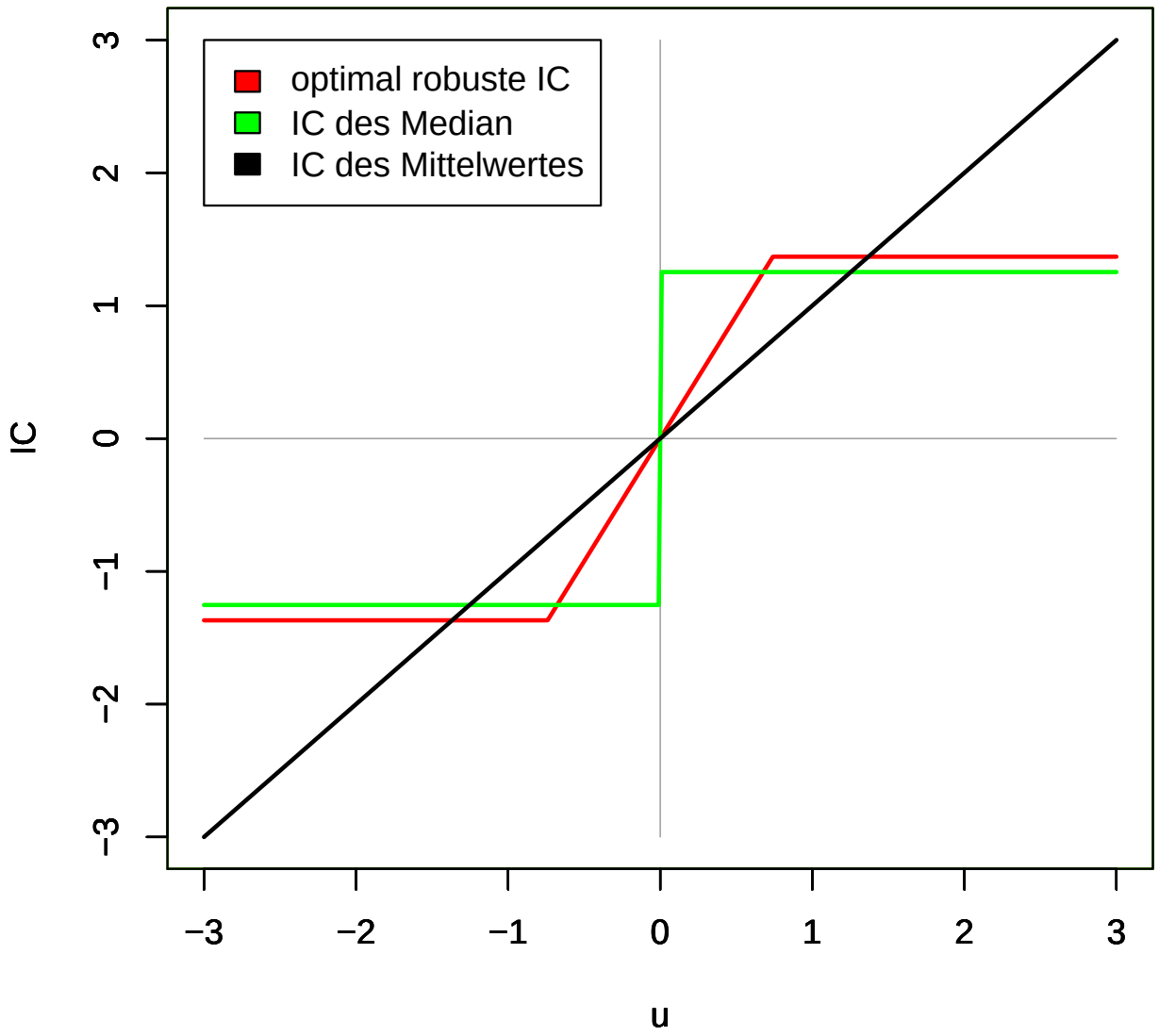
$$\mathrm{E}_{P_\theta}[\psi_\theta] = 0 \qquad (5.2.1)$$

$$\mathrm{E}_{P_\theta}[\psi_\theta \Lambda_\theta^\tau] = \mathbb{I}_k \qquad (5.2.2)$$

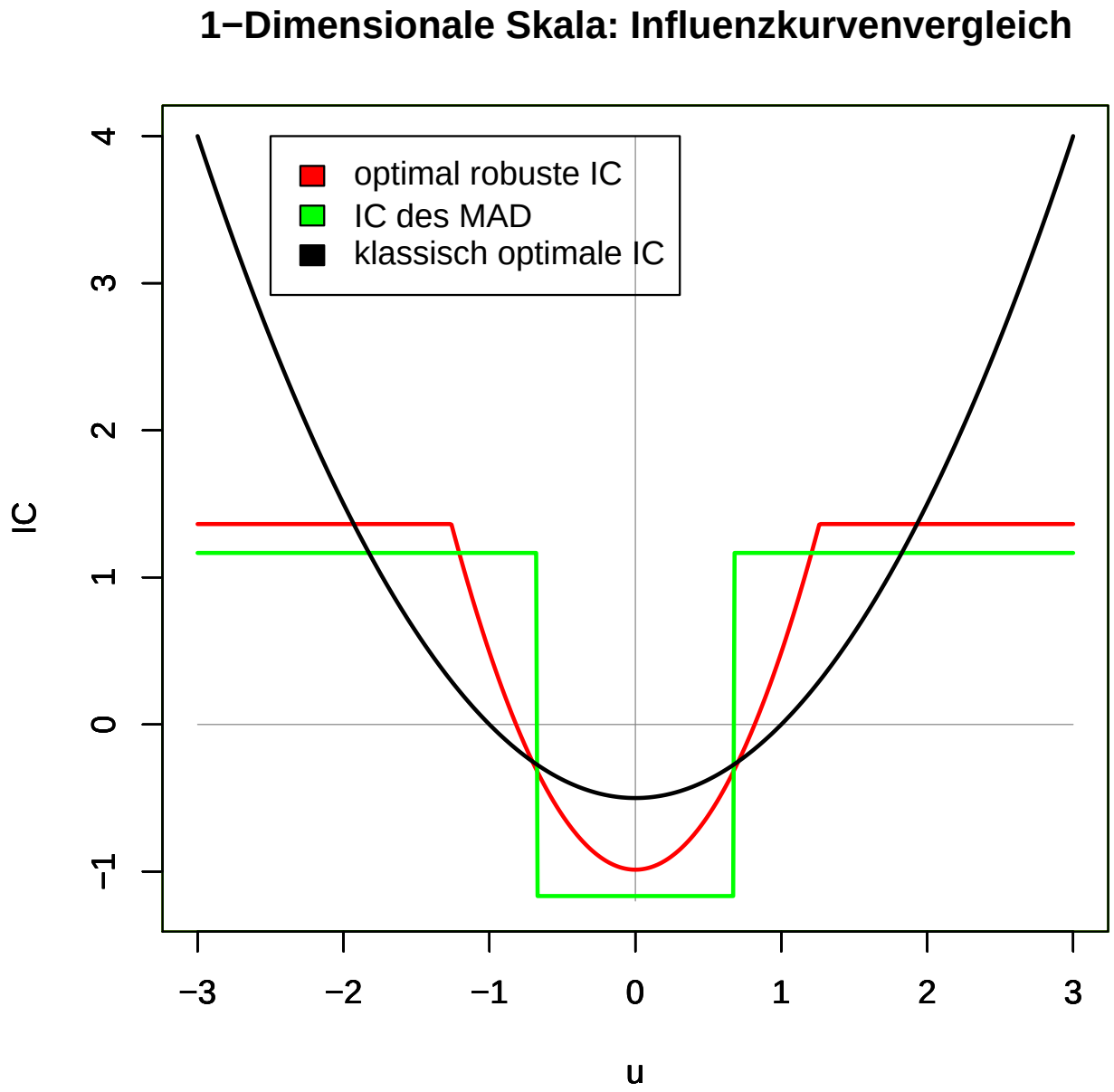
    dabei ist Λ_θ die L_2–Ableitung von P_θ, die sich bei Glattheit bestimmt als $\frac{\partial}{\partial\theta}\log p_\theta$; ein solches ψ heißt Influenzkurve für P_θ in θ

- relative Effizienz
  - bei Verwendung von robusten Verfahren wird man im allgemeinen gegenüber dem klassisch optimalen Verfahren — oft durch Maximum Likelihood zu bestimmen — im idealen Modell asymptotisch einen Verlust an Effizienz eingehen, indem die (Spur der) Varianz größer sein wird als beim optimalen Verfahren
  - Anscombe: Betrachte diesen Effizienzverlust als Versicherungsprämie, der für einen Schutz gegen Ausreißer zu bezahlen ist
  - Beispiel: $\mathrm{relEff.verl}(\mathrm{Med} : \bar{X}_n) \doteq 57\%$
  - Interpretation: um mit dem Median im idealen Modell die gleiche Genauigkeit wie mit dem arithmetischen Mittel zu erreichen, brauche ich 57% mehr Beobachtungen

- Lokation — $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, 1)$, gesucht μ
  - klassisch optimal: $\bar{X}_n$
  - am robustesten : $\text{med}\{X_i\}$ (Bruchpunkt 50%)
  - minimiert maximalen MSE in einer 10%–Kontaminationsumgebung für $n = 36$:
    $\check{\theta} := \text{Median}(X_i)$, $\psi(x) := 1.848x \min\{1, \frac{0.741}{|x|}\}$,
    $\psi_\theta(x) = \psi(x - \theta)$, $\theta^{(0)} := \check{\theta} + \frac{1}{n}\sum_{i=1}^{n} \psi_{\check{\theta}}(x_i)$
  - auch erreichbar durch
    - M–Schätzer: Nullstelle θ_0 von
      $F(\theta; x_1, \ldots, x_n) := \sum_i \psi_\theta(x_i)$
    - L–Schätzer: getrimmtes Mittel mit $\alpha = 30.1\%$
    - R–Schätzer: auf Rängen basierender Schätzer


1−Dim. Lokation: Influenzkurvenvergleich
optimal robuste IC
IC des Median
IC des Mittelwertes
IC
u
−3
−2
−1
0
1
2
3

- Skala — $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0, \sigma^2)$, gesucht σ^2
  - klassisch optimal: $\frac{1}{n}\sum_{i=1}^{n} X_i^2$
  - am robustesten : MAD — $\text{med}\{|X_i - \text{med}\{X_j\}|\}$ (Bruchpunkt 50%)
  - lokale asymptotisch nicht zu unterscheiden vom IQR, aber dieser hat Bruchpunkt 25%
  - minimiert maximalen MSE in einer 10%–Kontaminationsumgebung für $n = 36$:
    $\check{\sigma} = 1.4826\text{MAD}(X_i)$, $Y_i := (X_i/\check{\sigma})^2$,
    $\theta^{(0)} := \check{\sigma} + \frac{\check{\sigma}}{n}\sum_{i=1}^{n} 1.4776(Y_i - 0.6680)\min\{1, \frac{0.9220}{|Y_i - 0.6680|}\}$


1−Dimensionale Skala: Influenzkurvenvergleich
optimal robuste IC
IC des MAD
klassisch optimale IC
IC
u
−3
−2
−1
0
1
2
3
−1
0
1
2
3
4

5.2.3 (d) robuste Verfahren in R

- Lokation/Skala: in Bibliothek MASS: huber, hubers
- Regression: in Bibliotheken MASS und lqs: rlm; verwendete / implementierte Verfahren:
  - Startschätzer: `LTS` (Least trimmed squares), S–Schätzer
  - ψ–Funktionen: Huber–, Hampel– und Bisquare
- viel mehr (extra Bibliothek) in `S-Plus`

5.2.3 (e) Literatur

- Einstieg: Huber (1977)
- weitergehend: Huber (1981), Hampel et al. (1986), Rieder (1994)

6 numerische Algorithmen in S-Plus/R

6.1 Interpolation

6.1.1 Problemstellung

- gegeben eine Funktion $x \mapsto f(x)$ auf einem $x, f(x)$–Gitter $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$
- gesucht der Funktionswert $f(x_0)$ an einer Stelle $x_0 \notin \{x_1, \ldots, x_n\}$

6.1.2 Methoden

- Polynominterpolation
  - Idee: Funktion verhält sich lokal wie ein Polynom

- Tatsache: auf Kompakta liegen die Polynome dicht im $L_p(\mu)$, μ irgendein endliches Maß auf $\mathbb{B}^k$ und $p \in [1, \infty)$
- einfachstes Beispiel: lineare Interpolation
  gegeben (x_1, y_1), (x_2, y_2), $x_1 < x_0 < x_2$;
  Schätzung für y_0 durch
  $\hat{y}_0 := y_2(x_0 - x_1)/(x_2 - x_1) + y_1(x_2 - x_0)/(x_2 - x_1)$
- allgemein: Lagrange–Polynome
  für n Stützstellen (x_i, y_i) verwendet man die
  Lagrange–Polynome $L_{i;\,x_1,\ldots,x_n}(x)$ mit $L_i(x_j) = 0$ für $i \neq j$
  und $L_i(x_i) = 1$ und erhält als interpolierendes Polynom
  $\hat{f}_{x_1,\ldots,x_n}(x) := \sum_i y_i L_i(x)$ — genaueres siehe Stoer (1999)

- rationale Interpolation
  - Idee: statt Polynomen zur Interpolation werden rationale Funktionen, i.e. Quotienten von Polynomen zur Interpolation verwendet
  - oft auch zur Modellierung einer Asymptotik für $|x| \to \infty$,

falls diese vom Typ αx^{-k} für ein $k \in \mathbb{N}$

- Spline–Interpolation
  - möchte möglichst "glatt" durch die Interpolations–Punkte
  - "Glattheit" $\hat{=}$ Größe der zweiten Ableitung $\Rightarrow$ Idee: minimiere $\int_{x_1}^{x_n} (f''(x))^2\, dx$ unter allen interpolierenden Funktionen
  - Lösung: kubische Splines
- nicht 100%-ige Datentreue
  - sind die Meßwerte y_i nicht vollständig zuverlässig, kann man auch zu gunsten einer "einfacheren" (oder glatteren) Interpolationsfunktion zulassen, dass $\hat{f}(x_i) \neq y_i$
  - führt auf kombinierte Datentreue / Glattheit (niedrige Ordnung) — Kriterien
  - *Glättungskurven*, *Glättungssplines*

6.1.3 Gütekriterien

- generell Datentreue (zum Beispiel gemessen in $\sum_i |f(x_i) - y_i|^2$)
- bei Polynominterpolation: Einfachheit, also möglichst geringer Grad des interpolierenden Polynoms
- bei Splines: Glattheit gemessen in $\int_{x_1}^{x_n} (f''(x))^2 \, dx$

6.1.4 Vor– und Nachteile

+ oft einfach zu bestimmen, einfacher als tatsächliche Funktion f
– prinzipiell schlecht bei Extrapolation
? Behandlung von Bindungen — also mehrere Auswertungen an einer x–Stelle

6.1.5 Umsetzung in R

- Lineare / Konstante Interpolation: die Funktionen **approx** und **approxfun**
  - Syntax:

```
approx(x, y, xout, method="linear", n=50,
       yleft, yright, rule = 1, f=0,
       ties = mean)
approxfun(x, y, method="linear",
          yleft, yright, rule = 1, f=0,
          ties = mean)
```

  - Argumente
    * `x`,`y`: Interpolationsgitter
    * `xout`: (optional) Vektor mit Auswertungsstellen
    * `method`: Interpolationsmethode — "linear" oder "constant"

* `n`: falls keine Auswertungsstellen in `xout` spezifiziert werden, wird auf einem äquidistanten Gitter von `n` im Intervall [**min**(x),**max**(x)] interpoliert
* `yleft`,`yright`: Wert der zurückgegeben wird, falls die Auswertungsstelle `x` kleiner (größer) als **min**(x) (**max**(x)) ist; per default durch das Argument `rule` festgelegt
* `rule`: legt fest, was als Extrapolationsverfahren verwendet werden soll; `1` $\hat{=}$ NAs werden zurückgegeben, `2` $\hat{=}$ wird der Gitterwert an der Gitterstelle zurückgegeben, die am nächsten zur Auswertungsstelle liegt.
* `f`: Falls `method="constant"` eine Zahl aus $[0, 1]$, die einen Kompromiss zwischen links– und rechtsstetiger Treppenfunktion angibt. Sind die Gitterwerte links und rechts der Auswertungsstelle `y0` und `y1`, so wird `y0*(1-f)+y1*f` zurückgegeben.
* `ties`: Behandlung von Bindungen (nicht hier!)

- Details
  - Es werden nur vollständige Paare (x, y) zur Interpolation verwendet
- Rückgabewert
  - **approx** gibt eine Liste mit Komponenten x und y, zurück — die n Interpolationspunkte
  - **approxfun** gibt eine Funktion zurück, die dann lineare / konstante Interpolation durchführt; zu einem gegebenen x–Vektor als Argument gibt diese (zurückgegebene) Funktion die entsprechenden Interpolationswerte zurück

R–Beispiel 6.1-1 [lineare/konstante Interpolation]:

```
# Gitter der Fkt-werte
 x ← 1:10; y ← rnorm(10)
#Plot der Gitterpunkte
par(mfrow = c(2,1))
plot(x, y,
 main = "approx(.) and approxfun(.)")
points(approx(x, y), col = 2, pch = "*")
points(approx(x, y, method = "constant"),
       col = 4, pch = "*")
# interpolierende Kurve/Konstante
f ← approxfun(x, y)
curve(f(x), 0, 10, col = "green")
points(x, y)
is.function(fc ← approxfun(x, y,
                method = "const")) # TRUE
```


```
curve(fc(x), 0, 10, col = "darkblue",
                    add = TRUE)
```

- (kubische) Spline–Interpolation: die Funktionen **spline** und **splinefun**
  - Syntax:

```
spline(x, y = NULL, n = 3*length(x),
       method = "fmm", xmin = min(x)
       xmax = max(x))
splinefun(x, y = NULL, method = "fmm")
```

  - Argumente
    - `x`,`y`: Interpolationsgitter
    - `xout`: (optional) Vektor mit Auswertungsstellen
    - `n`: es wird auf einem äquidistanten Gitter von `n` im Intervall [**min**(x),**max**(x)] interpoliert
    - `method`: Spline–Interpolationsmethode — `"fmm"`, `"natural"` oder `"periodic"`

    * xmin,xmax: linker und rechter Endpunkt des Interpolationsbereichs
- siehe auch
    * Package splines
    * Funktion smooth.**spline** im Package modreg
    * Contributed Packages akima, cobs, gss, pspline
- Rückgabewert
    * **spline** gibt eine Liste mit Komponenten x und y, zurück — die n Interpolationspunkte
    * **splinefun** ergibt wie **approxfun** eine Funktion
- R–Beispiel 6.1-2 [kubische Spline–Interpolation]:

```
#Graphikvorbereitung
op <- par(mfrow = c(2,1), mgp = c(2,.8,0),
          mar = .1+c(3,3,3,1))
#Gitter
n <- 9; x <- 1:n; y <- rnorm(n)
```


```
plot(x, y, main =
  paste("spline[fun](.) through",
        n, "points"))

lines(spline(x, y))
lines(spline(x, y, n = 201), col = 2)
#eine glaettere Funktion
y ← (x-6)^2
plot(x, y, main = "spline(.) -- 3 methods")
lines(spline(x, y, n = 201), col = 2)
lines(spline(x, y, n = 201,
      method = "natural"), col = 3)
lines(spline(x, y, n = 201,
      method = "periodic"), col = 4)
legend(6,25, c("fmm","natural","periodic"),
       col=2:4, lty=1)
f ← splinefun(x, y)
```


```
ls(envir = environment(f))
splinecoef ← eval(expression(z),
        envir = environment(f))
curve(f(x), 1, 10, col = "green", lwd = 1.5)
points(splinecoef, col = "purple", cex = 2)
par(op)
```

6.2 numerische Invertierung

6.2.1 Problemstellung

- gegeben eine monotone Funktion $x \mapsto f(x)$ auf einem $x, f(x)$–Gitter $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$
- zu einem vorgegebenen Funktionswert f_0, $f_0 \notin \{f(x_1), \ldots, f(x_n)\}$ ist die Stelle x_0 gesucht, so dass $f(x_0) = f_0$

6.2.2 Methode

- vertausche x und y und verwende Interpolationsmethode

6.3 Integration

6.3.1 Problemstellung

- gegeben eine Funktion $f : \mathbb{R} \to \mathbb{R}$ und ein Intervall $[a, b]$
- gesucht $\int_a^b f(x)\,dx$

6.3.2 Methoden

- Trapezverfahren, Simpsonverfahren
- Schrittweitenwahl
- Gaußintegration
- Extrapolationsverfahren

6.3.3 Umsetzung in R

- die Funktion integrate
  - Syntax

```
integrate(f, lower, upper,subdivisions=100,
          rel.tol = .Machine$double.eps^0.25,
          abs.tol = rel.tol, stop.on.error = T,
          keep.xy = FALSE, aux = NULL, ...)
```

  - Argumente
    * f: eine R–Funktion $\mathbb{R} \to \mathbb{R}$, die als erstes Argument die Integrationsvariable x hat und als Rückgabewert einen Vektor derselben Länge wie x; bei Rückgabe von $\pm\infty$ Fehler
    * lower, upper: Integrationsgrenzen; dürfen unendlich sein
    * subdivisions: Maximalzahl an Gitterpunkten
    * rel.tol, abs.tol: relative / absolute geforderte Genauigkeit

    * `stop.on.error`: (logisch) soll bei Fehlern abgebrochen werden (default) oder eine Warnmeldung herausgegeben werden?
    * `keep.xy`, `aux`: nicht benutzt aus Kompatibilitätsgründen mit `S`
    * `...`: weitere Argumente für `f`
  - Details
    * ist mindestens eine der beiden Grenzen unendlich, wird das unbegrenzte Intervall auf ein begrenztes abgebildet
    * implementiert ist ein global adapitives (Schrittweitenwahl!) Verfahren zusammen mit einer Extrapolationstechnik
    * basiert auf `QUADPACK`–Routinen `dqags` und `dqagi` von R. Piessens und E. deDoncker-Kapenga, aus der `Netlib`–Bibliothek
    * rel.tol kann nicht weniger als **max**(50∗.**Machine$double**.eps, 0.5e−28) sein, sofern **abs**.tol <= 0

- Rückgabewert: eine Liste vom Typ/Klasse `"integrate"` mit Attributen
  - `value`: numerische Näherung für das Integral
  - `abs.error`: Abschätzung für den Betrag des absoluten Fehlers
  - `subdivisions`: tatsächliche Zahl an Stützstellen
  - `message`: `"OK"` oder eine Warnmeldung
  - `call`: der Funktionsaufruf

- die Funktion adapt aus dem CRAN–Paket adapt
  - integriert eine Funktion $f : \mathbb{R}^d \to \mathbb{R}$ über ein d-dimensionales Rechteck $[a, b]$, also $\int_{[a,b]} f \, d\lambda^d$´, wobei $2 \leq d \leq 20$;
  - unbeschränkte Intervalle sind nicht erlaubt
  - Syntax

```
adapt(ndim, lower, upper, minpts = 100,
      maxpts = NULL, functn, eps = 0.01, ...)
```

  - Argumente
    * ndim: $\hat{=}d$ – die Dimension des Integrationsbereichs
    * lower, upper: Vektoren der Länge ndim mit den unteren (oberen) Integrationsgrenzen
    * minpts, maxpts: minimale / maximale Zahl an Stützstellen; maxpts per default NULL
    * functn: Integrand; siehe Parameter f in integrate
    * eps: gewünschte Genauigkeit für den relativen Fehler
    * ...: weitere Übergabeparameter für functn
  - Details

* modifizierter Code von Mike Meyer's S Code; rufen eine FORTRAN subroutine von A.C. Genz auf
* die Original FORTRAN Funktion ist zur Kompatibilität mit R auf doppelte Genauigkeit modifiziert
* arbeitet nur für 2 oder mehr Dimensionen, in einer Dimension wird integrate aufgerufen
* indem man maxpts auf NULL setzt, verdoppelt adapt beginnend mit max(minpts,500, r(ndim)) in jedem Schritt maxpts, bis die gewünschte Präzision erreicht ist oder der Speicher nicht mehr ausreicht;

- man beachte, dass die Zahl der nötigen Gitterpunkte typischerweise exponentiell mit der Dimension `ndim` wächst und der zugrundeliegende Code mindestens `maxpts >= r(ndim)` verlangt, wobei $r(d) = 2^d + 2d(d+3) + 1$
- Rückgabewert: eine Liste vom Typ/Klasse `"integration"` mit Attributen
  * `value`: numerische Näherung für das Integral
  * `relerr`: Abschätzung für den relativen Fehler;
  * `minpts`: tatsächliche Zahl der Stützstellen
  * `ifail`: Fehlerindikator; fall `ifail` nicht `0` ist, gibt es einer Warnung

6.4 Lösen von Gleichungssystemen II

6.4.1 Problemstellung

- Gegeben n Gleichungen (über $\mathbb{R}$) in n Unbekannten $\Longrightarrow$ $F : \mathbb{R}^n \to \mathbb{R}^n$ und vorgegebene rechte Seite b
- gesucht $x_0 \in \mathbb{R}^n$ mit $F(x_0) = b$

6.4.2 Methoden

- Spezialfall: F linear $\rightsquigarrow$ F gegeben als $F(x) = Ax$;
  in R x0←**solve**(A,b)
  - im allgemeinen mit `QR` oder `SVD`–Zerlegung
  - spezielle Verfahren für strukturiertes A
    - (band–)diagonal
    - obere Dreiecksmatrix
    - dünn besetzt
  - in R: das *Contributed Package* `Matrix`

- stochastische Suche:
  ziehe zufällig zulässige x–Werte und wähle das x_0 mit minimalem $|F(x_0) - b|$
  - ? Frage: mit welcher W-keit?
  - \+ sehr einfach zu realisieren, keine Struktur nötig
  - \+ Konsistenz: mit beliebig hoher W–keit komme ich schließlich dem wahren x_0 (so es existiert) bis auf ein vorgegebenes ε nahe
  - – sehr langsam
  - – keine determistische Lösung
  - – Variante: die Ziehungswahrscheinlichkeit sich um die bisher besten Werte konzentrieren lassen
    - ? mit welcher Rate?
    - ∗ Schlagworte: Sintflut–Algo, Simulated Annealing, genetische Algorithmen

- bei Monotonie und $n = 1$: Bisektion

  [-1] finde Startintervallgrenzen x_l und x_r
  bestimme Abbruch–ε, Maxzahl an Iterationen maxit

  [0] setze $i := 0$, $\mathrm{Fehler} := \varepsilon + 1$

  [1] $i := i + 1$, $x := (x_r + x_l)/2$, $f := F(x)$, $\mathrm{Fehler} := |f - b|$

  [2] falls F steigt: { falls $f < b$ $x_l := x$, sonst $x_r := x$ }
  falls F steigt: { falls $f < b$ $x_r := x$, sonst $x_l := x$ }

  [3] gehe zu $[1]$ solange $i < \mathrm{maxit}$ und $\mathrm{Fehler} > \varepsilon$

  \+ sehr einfach zu realisieren

  \+ geometrische Konvergenz (Intervalllänge wird exponentiell — aber mit fester Rate kürzer)

  – lokal gibt es u.U. bessere Verfahren

  – klebt an der Monotonie

- bei Glattheit: Newtonverfahren

  [-1] Bestimme $f(x) = F'(x)$; finde Startwert x_1;
  bestimme Abbruch–ε, Maxzahl an Iterationen maxit, — evtl. Schrittweiten–Begrenzung

  [0] setze $i := 0$, $\mathrm{Fehler} := \varepsilon + 1$

  [1] $i := i + 1$, $x := x - (f(x))^{-1} F(x)$, $f := F(x)$,
  $\mathrm{Fehler} := |f - b|$

  [2] gehe zu $[1]$ solange $i < \mathrm{maxit}$ und $\mathrm{Fehler} > \varepsilon$

  \+ lokal quadratische Konvergenz (unglaublich schnell!)

  – ab wann sind wir "lokal"?

  – unter Umständen sehr instabil (für $f(x_0) \approx 0$)

  – benötigt Ableitung

- bei Kontraktion: Fixpunktverfahren
  - dazu: $G(x) = -\operatorname{sign}(\text{“}F'(x)'')(F(x) - b) + x$
  - gesucht x_0 mit $G(x_0) = x_0$
  - falls $|G'(x)| < 1$ bzw. im eindimensionalen
    $\Longleftrightarrow$ “$|F'(x)|'' < 2$ ($\hat{=}$ *Kontraktion*) in einer Umgebung von x_0 und Startwert x_1 so, dass die Folge der x_k nie diese Umgebung verlässt, herrscht Konvergenz — Banachscher Fixpunktsatz

[-1] finde Startwert x_1;
bestimme Abbruch–ε, Maxzahl an Iterationen maxit

[0] setze $i := 0$, $\mathrm{Fehler} := \varepsilon + 1$

[1] $i := i + 1$, $f := G(x)$, $\mathrm{Fehler} := |f - x|$, $x := f$,

[2] gehe zu $[1]$ solange $i < \mathrm{maxit}$ und $\mathrm{Fehler} > \varepsilon$

---

genügt entsprechende Lipschitzbedingung

! besser als Bisektion, falls $0.5 < |\text{“}F'(x)''| < 1.5$

\+ geometrische Konvergenz — sofern x_k in $\{x : |G'(x)| < 1\}$ bleiben

\+ benötigt keine Monotonie, funktioniert auch mehrdimensional

\+ benötigt keine Ableitung — nur lokale Lipschitzbedingung an F

– wann ist das der Fall?

– u.U. auch langsamer als Bisektion

– lokal gibt es u.U. viel bessere Verfahren

6.4.3 Literatur

- Press et al. (1992)
- Stoer (1999)

6.4.4 Umsetzung in R

- **uniroot**
- **polyroot**
- siehe auch Abschnitt "Zero–Finding", im Manual "Writing R–Extensions", pp40–41
- Schnittstelle zu den Numerical Recipes (Press et al. (1992)):

  http://lib.stat.cmu.edu/S/recipes

6.5 Minimierung

6.5.1 Problemstellung

- Gegeben eine Funktion $F : \mathbb{R}^n \to \mathbb{R}$
- gesucht $x_0 \in A \subset \mathbb{R}^n$ mit $F(x_0) = \min_{x \in \mathbb{R}}$

6.5.2 Klassen von Problemen

- Definitionsbereich
  - diskret $\leftrightarrow$ kontinuierlich
  - Dimension
  - Restriktionen: mit oder ohne Nebenbedingungen
    - Gleichheitsnebenbedingungen
    - Ungleichheitsnebenbedingungen
- Funktion F
  - konvex oder nicht
  - differenzierbar oder nicht
  - unimodal oder nicht
  - speziell: linear oder quadratisch?

- Optimierungstechniken
  - kombinatorische Optimierung
    - Branch and Bound/Cut
    - stochastische Techniken
  - Lineare Optimierung
    - Simplex–Algorithmus
    - graphentheoretische Algorithmen
  - Quadratische Optimierung
  - glatte, konvexe Optimierung
    - globale Optimierung
      - stochastische Techniken
      - gemischt stochastisch – deterministisch
    - lokale Optimierung
      - Differentielle Argumente
      - diff–bare Lagrangetechniken

6.5.3 Methoden

- Minimierung auf Gitter
  - – "Brute Force" — sehr langsam
  - \+ sehr leicht durchzuführen
  - \+ keine Ableitungen nötig
  - ? wie repräsentativ ist das Gitter?
- stochastische Suche
  - – Eigenschaften siehe entsprechender Punkt bei Gleichungen
- bei Bitonie und $n = 1$:
  spezielles "Bisektions"–Verfahren —
  das *"Goldener–Schnitt–Verfahren"*

[-1] Setze $\alpha := 3/2 - \sqrt{5}/2$ und suche Startintervall $[x_l, x_r]$,
$x_m := x_l + \alpha(x_r - x_l)$, so dass $F(x_l) > F(x_m) < F(x_r)$;
bestimme Abbruch–ε, Maxzahl an Iterationen maxit

[0] setze $i := 0$, $\mathrm{Fehler} := \varepsilon + 1$, setze $s := 1$;

[1] $i := i + 1$,
falls $s = 1$ setze $x_t := x_m + \alpha(x_r - x_m)$,
sonst $x_t := x_l + \alpha(x_m - x_l)$; $f_j := F(x_j)$, $j = t, l, m, r$;
Fehler $:= \max_k |f_k - \min_j f_j|^2$

[2] falls $x_t < x_m$: $\Big\{$ falls $f_t < f_m$: $\{x_r := x_m,\ x_m := x_t,\ s := 0$ $\}$, sonst $\{\ x_l := x_t,\ s := 1\ \}$ $\Big\}$
falls $x_t > x_m$: $\Big\{$ falls $f_t < f_m$: $\{\ x_l := x_m,\ x_m := x_t,\ s := 1$ $\}$, sonst $\{\ x_r := x_t,\ s := 0\ \}$ $\Big\}$

[3] gehe zu [1] solange $i <$ maxit und Fehler $> \varepsilon$

  - \+ sehr einfach zu realisieren
  - \+ geometrische Konvergenz (Intervalllänge wird exponentiell — aber mit fester Rate kürzer)
  - – lokal gibt es u.U. bessere Verfahren
  - – klebt an der Bitonie
- im mehrdimensionalen: Trennung in Länge und Richtung $\rightsquigarrow$
  - – Line–Search (siehe oben)
  - – Richtungssuche:
    - $*$ steepest Descent (Gradienten–Verfahren)
    - $*$ (modifiziertes) Newtonverfahren
    - $*$ cg–Verfahren
- bei Glattheit: Lösungsverfahren für $F'(x) \stackrel{!}{=} 0$ aus Abschnitt 5.2.2 (c)

6.5.4 Literatur

- Press et al. (1992)
- Fletcher (1987), Luenberger (1969), 1984
- Geiger and Kanzow (1999), 2002

6.5.5 Umsetzung in R

- **optim**
- **nlm**
- **optimize**
- Schnittstelle zu den Numerical Recipes (Press et al. (1992)):

  http://lib.stat.cmu.edu/S/recipes.

6.6 sich selbst verändernde Programme

- Idee: lasse während des Ablaufs des Programms durch dieses R–code produzieren, der dann später abgearbeitet wird
- R–Beispiel 6.6-1 **eval** und **parse**:

  Code: hier

```
m<-10
n<-21
#####################################
#
### Dichte der m+1. Ordnungsstatistik
##
### beliebige Verteilung, zB: FUN="norm", "chisq",
#
dichte<-function(FUN=stop("keine Funktion"),
```


```
                    n=stop("kein n"),
                    x=stop("kein Argument"),...)
#
#Beachte Funktion kann mit zusaetzlichen
#Parametern --- wie df fuer chisq auf-
#gerufen werden!
#
#berechnet zu "belibigem" F (aus einer
#der implementierten Verteilungsklassen)
#f_{i:n}
#
#
#FUN: String mit Funktionsnamen
#n: Umfang der Stichprobe (=2m+1)
#x: Auswertungsstelle(n) der Dichte
#...: weitere Argumente,
#      z.B. df fuer chisq
```


```
#
#Rueckgabewert  f_{m+1:2m+1}(x)
#
{if (n%%2==0) stop("nur ungerades n")
  m<-n%/%2+1

  #Berechnung von i (n choose i)
  C0<-m*gamma(n+1)/gamma(m+1)/gamma(m)

  #Berechnung von F(x), f(x)
  eval(parse(text=
    paste("f1<-p",FUN,"(q=x,...)",sep="")))
  eval(parse(text=
    paste("d1<-d",FUN,"(x=x,...)",sep="")))

 di<-C0*f1^(m-1)*(1-f1)^(m-1)*d1
 return(di)
```


```
}
#
### Integranden fuer E[Median],
##                    Var[Median]
##
### beliebige Verteilung,
##  zB: FUN="norm", "chisq"
#
#FUN: String mit Funktionsnamen
#n: Umfang der Stichprobe (=2m+1)
#x: Auswertungsstelle(n) der Dichte
#...: weitere Argumente,
#      z.B. df fuer chisq
#
#Rueckgabewert x^i f_{m+1:2m+1}(x),
#i=1,2
```


```
dix <- function(x,FUN=stop("keine Funktion"),
                n=stop("kein n"),...)
 {return(x*dichte(x=x,FUN=FUN,n=n,...))}
dix2 <- function(x,FUN=stop("keine Funktion"),
                 n=stop("kein n"),...)
 {return(x^2*dichte(x=x,FUN=FUN,n=n,...))}


#
### Vergleich von tatsaechlicher Dichte und
### Dichte der as. NV
##  sowie tatsaechlicher und asymptotischer
##  Varianz
##
### beliebige Verteilung,
##  zB: FUN="norm", "chisq",...
#
dvgl <- function(FUN=stop("keine Funktion"),
```


```
                    n=stop("kein n"),
                    x=stop("kein Argument"),...)
#
#FUN: String mit Funktionsnamen
#n: Umfang der Stichprobe (=2m+1)
#x: Auswertungsstelle(n) der Dichte
#...: weitere Argumente,
#      z.B. df fuer chisq
#
#Rueckgabewert f^{(asy)}_{m+1:2m+1}(x),
#
{ #linker und rechter Integrations-Rand
   eval(parse(text=
    paste("li<-q",FUN,"(.0025,...)",sep="")))
   eval(parse(text=
    paste("re<-q",FUN,"(.9975,...)",sep="")))
```


```
#Median von FUN, Med(F)
eval(parse(text=
 paste("q0<-q",FUN,"(.5,...)",sep="")))

#Dichte an der Stelle Med(F)
eval(parse(text=
 paste("d0<-d",FUN,"(q0,...)",sep="")))

#asymptotische Dichte
di<-dnorm(x,mean=q0,sd=1/2/d0/sqrt(n))

#exakte Dichte
di0<-dichte(FUN=FUN,n=n,x=x,...)

#exakter Erwartungswert von Med_21
# -- numerisch integriert
me<-integrate(dix,li,re,
```


```
        FUN=FUN,n=n,...)$value

#exakte Varianz von Med_21
#-- numerisch integriert
va<-integrate(dix2,li,re,
        FUN=FUN,n=n,...)$value-me^2

#Aufskalierung mit n
van<-va*n

#Vergleich mit n
va.as<-(1/2/d0)^2
print(paste("Varianzen:␣",van,
      "(exakt)", va.as,"(asymptot)"))

#Plot der Dichten
matplot(x,cbind(di,di0),type="l")
```


```
    return(di)
}

#
### einige Auswertungen
#
x<-seq(-3,3,0.03)
d1<-dichte(FUN="norm",n=21,x=x)
d2<-dvgl(FUN="norm",n=21,x=x)

#
### exaktes Konfidenzintervall
#    minimaler Laenge
##
### beliebige Verteilung,
#    zB: FUN="norm", "chisq",...
#
```


```
#rechter Endpunkt bei vorgegebenem linken
#
repKI <- function(FUN=stop("keine Funktion"),
                  n=stop("kein n"), lip=-30,
                  itmax=40, del=10^(-8), alpha=0.05,
                  ll=-100, rr=100, ...)
#
#FUN: String mit Funktionsnamen
#n: Umfang der Stichprobe (=2m+1)
#lip: linker Aufpunkt des KI's
#itmax: maximale Zahl an Iterationen
#del: Abbruch--epsilon
#alpha: 1- Ueberdeckungs-WK des KI's
#ll: Linker Rand der Integration
#rr: Rechter Rand der Integration
#...: weitere Argumente,
```


```
#       z.B. df fuer chisq
#
#Rueckgabewert: rechter Endpunkt des KI's
#
{
# linker und rechter Startpunkt fuer
# Bisektion fuer rechten Endpunkt

   me<-integrate(dix,ll,rr,
          FUN=FUN,n=n,...)$value
   va<-integrate(dix2,ll,rr,
          FUN=FUN,n=n,...)$value-me^2
   rel<-me
   rer<-9*sqrt(va)+me

#  print(c(rel,rep))
```


```
# Bisektionsalgo
  i<-0
  de<-del+1
  while((i<itmax)&&(de>del))
        {rep0<-(rel+rer)/2
          i<-i+1
          we<-integrate(dichte,lip,rep0,
                  FUN=FUN,n=n,...)$value
#         print(c(i,we))
         if (we<1-alpha)
                  rel<-rep0
          else
                  rer<-rep0
          de<-abs(we-1+alpha)
         }
 if (we<1-alpha-del) stop("keine Überdeckung")
 return(rep0)
```


```
}

#Test
#
repKI(FUN="exp",n=21,ll=-30,rr=30)

#Bestimmung des KI minimaler Laenge
#
KIlr<-function(FUN=stop("keine Funktion"),
               n=stop("kein n"),lip=-30,
               itmax=40,del=10^(-8),alpha=0.05,
               step=1/10^4,ll=-100,rr=100,
               lir,lil,...)
#
#FUN: String mit Funktionsnamen
#n: Umfang der Stichprobe (=2m+1)
#lip: Startpunkt fuer linken Aufpunkt
```


```
#       des KI's
#itmax: maximale Zahl an Iterationen
#del: Abbruch--epsilon
#alpha: 1- Ueberdeckungs-WK des KI's
#ll: Linker  Rand der Integration
#rr: Rechter Rand der Integration
#lil: linkes Ende des
#       Linker-Endpunkt--Gitters
#lir: rechtes Ende des
#       Linker-Endpunkt--Gitters
#step: Schrittweite des
#        Linker-Endpunkt--Gitters
#...: weitere Argumente,
#       z.B. df fuer chisq
#
#Rueckgabewert: linker und rechter
#                  Endpunkt des KI's
```


```
#
{me<-integrate(dix,ll,rr,
        FUN=FUN,n=n,...)$value
 va<-integrate(dix2,ll,rr,
        FUN=FUN,n=n,...)$value-me^2
 if((missing(lir))||(missing(lil)))
    {lir<-me
     lil<--6*sqrt(va)+me
     }
 print(c(lil,lir) )
 la<-15*sqrt(va)
 rep0<-6* sqrt(va)
 i<-0
 for(li in seq(lil,lir,step))
     {reo<-rep0
        rep0<-repKI(FUN=FUN,n=n,lip=li,
                    itmax=itmax,del=del,
```


```
                    alpha=alpha, ll=ll,
                    rr=rr, ...)
        i <- i+1
        print(c(i, li, rep0, la,
                qnorm(.975)*2*sqrt(va)))
        lao <- la
        la <- rep0-li
        if (la>lao)
            break
      }
  return(c(li-step, reo))
}
#
### Beispiel-Auswertungen
#
Kllr(FUN="chisq",n=21,ll=-30,rr=30,
            step=.1,df=3)
```


```
KIlr(FUN="exp",n=21,ll=-30,rr=30)
#
```

- Vorgehensweise:
  (a) Zusammensetzen des Befehls-Strings mit Stringbefehlen wie **paste**
  (b) diesen in eine ausführbare `expression` mit **parse**`(text=<Befehl>)` wandeln
  (c) die `expression` auswerten mit **eval**

7 strukturierte Modelle

7.1 Regressionsmodelle

- Situation: zwei Variablen X und Y die irgendwie zusammenhängen;
  genauer: Y wird modelliert als Funktion von X und weiteren unbeobachtbaren Größen
- Problem: Rekonstruiere den Zusammenhang von X und Y
- Regressionsmodell: $Y = f(X, \varepsilon)$ mit
  - Fehlern ε
  - Beobachtungen Y
  - Regressoren X
  - Regressionsfunktion f
  - der Statistiker "sieht" (X, Y)

- weitere nötige Spezifikationen
  - Fehlerverteilung $\mathcal{L}(\varepsilon)$ — sind diese unabhängig, identisch verteilt?
  - stochastische $\leftrightarrow$ deterministische Regressoren
  - Spezifikation einer Klasse aus der f stammt —
    - parametrische Regression
      - lineare Parametrisierung?
    - nichtparametrische Regression
      - Glattheitsklasse
      - globale / lokale Bandweite

7.1.1 Lineare Statistische Modelle

Bemerkung 7.1-1:
Beachte: Nur die Parametrisierung ist linear, die Regressoren brauchen nicht linear einzugehen!

7.1.1 (a) Modellformulierung

- Matrixform $Y = X\theta + \varepsilon$, mit $Y = (Y_1, \ldots, Y_n)^\tau \in \mathbb{R}^n$, $X = (X_1^\tau, \ldots, X_n^\tau)^\tau$, $X_i \in \mathbb{R}^p$, $\theta \in \mathbb{R}^p$, $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)^\tau \in \mathbb{R}^n$
- mit Indizes $Y_i = X_i^\tau \theta + \varepsilon_i$,

Dabei ist / sind

- Y_i die i-te Beobachtung (abhängige Variable / Response)
- $X_i = X_{i,j}$ der Vektor der Regressoren (unabhängigen Variablen, Kovariate) zur i-ten Beobachtung
- $\varepsilon_i \overset{\text{u.i.v.}}{\sim} F$, $X_i \overset{\text{u.i.v.}}{\sim} K$ und unabhängig von ε_j oder X_i bekannt
- weiter sei $\operatorname{rk} X^\tau X = p$ bzw. $\operatorname{rk} \mathrm{E}[X_1 X_1^\tau] = p$

Beispiel 7.1-2 [ein Regressionsmodell]:

- Wir wollen den `hill`–Datensatz aus R analysieren;
- Datenbestand:
  - $Y \hat{=}$ Rekordzeiten zu verschiedenen Bergrennen
  - $X_{\cdot,1} \hat{=}$ Wegstrecke des Rennens in Meilen
  - $X_{\cdot,2} \hat{=}$ zu durchlaufende Höhenmeter
- Modell

$$Y_i := \theta_1 X_{i,1} + \theta_2 X_{i,2} + \theta_3 + \varepsilon_i,$$
$$\varepsilon_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0, \sigma^2)$$

weitere Referenz zu diesem Thema:

- Faraway (2002), im WWW verfügbar!,
- Christensen (1996)

7.1.1 (b) Formulierung in R

- Syntax: `<Resp> ~ <Regr1> + .... + <Regrp>` mit `<Resp>` der Response–Variablen und `<Regr1>`, ...,`<Regrp>` den Regressoren
- R–Beispiel 7.1-3 [das Modell aus Beispiel 7.1-2]:

```
library(MASS)
data(hills)
attach(hills)
hills
formula<-time~dist+climb
formula
```

- per default wird ein y–Achsenabschnitt mit eingepasst ($\hat{=}$ Regressor $\equiv 1$)
- soll kein y–Achsenabschnitt mit eingepasst werden, so gibt man dies durch Hinzufügen von `-1` auf die rechte Seite der Formel an

- bei Faktoren (diskreten Regressoren) wird per default für jedes Faktorniveau ein neuer Parameter / neuer y–Achsenabschnitt mit eingepasst
- innerhalb einer Formel: übliche arithmetische Ausdrücke möglich — z.B. für das lineare (!) Modell $y = \alpha \min(X_1, X_2) + \varepsilon$: `Y~pmin(X1,X2)`
- polynomiale Fits durch **poly**`(<Variable>,<Grad>)`, z.B. `Y~poly(X,2)` für $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$
- um `+`, `*` arithmetisch zu interpretieren: verwende `I`; so fittet `y~x+I(u^2+z)` das Modell mit Regressoren x und $u^2 + z$
- soll nur eine bestehende Formel `t` modifiziert werden, so kann man sowohl die linke als auch die rechte Seite der Modelldefinition durch `.` abkürzen $\rightsquigarrow$ **update**

7.1.1 (c) kategorielle Merkmale und ANOVA

Situation

- habe kategorielle Merkmale ($\hat{=}$ Faktoren) als Regressoren
- für lineares Modell sinnvoll: nur Regressoren $\mathrm{I}_{\{\mathrm{Faktor}=\mathrm{Wert}\}}$
- Zahl der Faktoren $n \hat{=}$ "n–fach–" oder "n–Wege–"Design
- sehr viele Parameter:
  Hat man n kategorielle Merkmale mit jeweils m_i Ausprägungen, ergeben sich $\prod_{i=1}^{n} m_i$ Parameter im "vollständigen" Design

Versuchsplanung

- kann die Versuchsbedingungen beim Versuchs–Design wählen
- um nicht zu viele Regressoren zu erzeugen, selten das vollständige Design — Problem multiplikativer Designs
- Planung des Versuchs: Welche Effekte sollen modelliert werden?

Arbeit mit Faktoren

- Einzel–, Kreuzeffekte und hierarchische Modelle
  - aus obengenanntem Dimensionsproblemen, beschränkt man sich darauf nur bestimmte Merkmalskombinationen zu betrachten
  - am häufigsten ordnet man jedem Merkmal einen Einzeleffekt zu — in R einfach durch Angabe der einzelnen Variablen durch $+$ verbunden in der Formel
  - daneben aber auch Kreuz– oder Interaktionseffekte — z.B. Variable X_1 mit Variable X_2, oder die Variablen X_1, X_4, X_6;

dies geschieht in R durch Angabe der Variablengruppe, in der die Gruppenelemente durch : getrennt werden, z.B: X1:X4:X6

- eine Abkürzung für X1+X4+X1:X4 ist X1*X4
- schließlich ist es auch möglich mit Potenzen zu arbeiten —
  - so bedeutet die Formel (1+a):(1+b):(1+**c**)−1 die Formel a+b+**c**+a:b+a:**c**+b:**c**+a:b:**c**
  - und die Formel (a+b+**c**)^2 (a+b+**c**):(a+b+**c**)
- sehr wichtige Technik zur Reduktion der Parametervielfalt: *hierarchische (verschachtelte) Modelle*
  - Beispiel: angenommen eine kategorielle Variable ist Geschlecht; demgemäß wird die Population aufgeteilt in männlich / weiblich und für jede der Teilpopulationen ein separates Modell (mit gleichen weiteren Kovariaten) eingepasst
  - in R realisiert durch `a / <Formel>`, mit `a` der Aufteilungs–Variablen und `<Formel>` der Formel, die in Abhängigkeit der Werte von `a` eingepasst werden soll

- Faktorkodierung
  - zur Umsetzung von kategoriellen / ordinalen Merkmalen in Kovariate muss man diese erst kodieren
  - normalerweise benötigt man für eine Variable mit m Merkmalsausprägungnen $m-1$ Indikator– oder *Dummy*–Variablen
  - diese Kodierungen ergeben die *Kontrastmatrix*:
    so erhält man mit der *Treatment–Codierung* für eine Variable mit 4 Ausprägungen drei Indikator–Variablen X_1, X_2, X_3;
    trägt man die Wertebelegung matrixwertig ab, so erhält man eine Matrix mit Zeilen entsprechend den Ausprägungen und Spalten entsprechend den Indikator–Variablen; in unserem

Beispiel

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

– folgende Kodierungen stehen in R zur Verfügung
  * default für kategorielle Variablen in R:
    Treatment–Codierung, i.e. ein Null–Niveau wird ausgezeichnet und jeder andere erhält genau für einen Indikator die 1, jeder Indikator enthält auch nur genau eine 1 — von Venables and Ripley (1999) empfohlen
    in R umgesetzt als **contr**.treatment
  * default für kategorielle Variablen in `S-Plus`:
    Helmert–Codierung, i.e. zusätzlich verlangt man noch, dass die Indikatoren auf der Variable für den y–Achsenabschnitt senkrecht stehen; schwieriger zu

interpretieren — in R umgesetzt als **contr.helmert**

- default für ordinale Variablen in R und `S-Plus`: *orthogonale Polynome* — R umgesetzt als **contr.poly**
- weiterhin implementiert: *Summen–Codierung* für ordinale Variablen, i.e. alle Einträge eines Indikators summieren sich auf 0
  in R umgesetzt als **contr.sum**

Formulierung von Designs in R

- als Konstruktionsmehtode sind in R nicht so viele Möglichkeiten gegeben wie in `S-Plus`;
- immerhin gibt es **expand.grid**

7.1.1 (d) Modell–Einpassung in R

- Methode: kleinste Quadrate
- die Funktion **lm**

- Syntax:

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

- Argumente:
  - `formula`: ein Formelausdruck wie in Abschnitt 7.1.1 (b) beschrieben
  - `data`: (optional) Data–Frame, der die Daten zum Fit des Modells enthält; per default werden die Daten der Umgebung `environment(formula)`, entnommen
  - `subset`: (optional) Vektor spezifiziert Teilstichprobe, mit der das Modell gefittet werden soll
  - `weights`: (optional) ein Vektor mit Gewichten, mit denen die einzelnen Residuen beim Fit gewichtet werden sollen; falls spezifiziert, wird ein gewichteter Kleinste–Quadrate–Fit

produziert; sonst werden gewöhnliche Kleinste Quadrate verwendet;

- `na.action`: eine Funktion, die bestimmt, wie mit Beobachtungen, die Missings (`NA`s) enthalten, verfahren werden soll. default: das Attribut `na.action` von `options` — falls dieses gesetzt ist, sonst `na.fail`, also Abbruch mit Fehlermeldung; die "Ur"–Einstellung ist `na.omit`.
- `method`: die Methode, die zum Fitten verwendet werden soll; z.Z. steht nur `method="qr"` zur Verfügung; `method="model.frame"` gibt den Modell–Frame zurück (das gleiche passiert mit `model = TRUE`, s.u.)
- `model`, `x`, `y`, `qr`: (logisch) falls `TRUE`, werden die entsprechenden Komponenten des Fits (i.e. der Modell–Frame, die Modellmatrix, die Beobachtungen, die `QR`–Zerlegung) mit ausgegeben
- `singular.ok`: (logisch): soll bei $\operatorname{rk} X < p$ weitergemacht werden? — per default `TRUE`. `FALSE` noch nicht

implementiert

  - `contrasts`: (optional) eine Liste mit den Kontrasten (siehe Abschnitt zur Arbeit mit Faktoren)
  - `offset`: Mittelwertkorrektur; kann benützt werden, um eine im Vornhinein bekannte Komponente in die lineare Vorhersage einzuschließen; kann natürlich auch in die Modell–Formel eingeschlossen werden; falls beides angegeben ist, wird die Summe der beiden Terme benutzt
  - `...`: weitere Argumente, die an weitere aufgerufene Regressions–Fit–Funktionen übergeben werden (s.u.)

- Details:
  - die Modelle für `lm` werden wie in Abschnitt 7.1.1 (b) und im Abschnitt zur Arbeit mit Faktoren spezifiziert; typischerweise von der Form `response ~ terms`
  - der `terms`–Teil der Formel kann auch als ein linearer Prädiktor für `response` aufgefasst werden

  - `lm` ruft selbst Hilfsfunctionen wie `lm.fit`, etc. auf , um die tatsächlichen Berechnungen durchzuführen
  - für eine ANOVA schätzt man erst das lineare Modell mit **lm** und verwendet dann die generische Funktion **anova** auf den Rückgabewert an
  - stattdessen für ANOVA auch möglich: direkt **aov**
- Rückgabewert: eine Datenstruktur vom Typ `"lm"` oder mehrere Antworten vom Typ `c("mlm", "lm")` mit Elementen
  - `coefficients`: Koeffizientenvektor ($\hat{=}$ Vektor der Parameterschätzungen); die Elemente tragen Namen
  - `residuals`: die Residuen, i.e. tasächlicher minus gefitteter Wert für Y
  - `fitted.values`: gefittete Beobachtungen
  - `rank`: Rang der Beobachtungsmatrix
  - `weights`: (nur für gewichtete Kleinste Quadrate) die angegebenen Gewichte

  - `df.residual`: Zahl der Freihheitsgrade der Residuen
  - `call`: der Aufruf, mit dem der Fit durchgeführt wurde
  - `terms`: die verwendete rechte Seite der Formel
  - `contrasts`: (nur falls relevant) die verwendeten Kontraste
  - `xlevels`: (nur falls relevant) die Merkmalsausprägungen der Faktoren, die zum Fit benützt worden sind
  - `y`: falls gewünscht, die benützten Beobachtungen
  - `x`: falls gewünscht, die benützte Modell– (oder Daten–)Matrix
  - `model`: falls gewünscht (per default: ja), der benützte Modell–Frame
- generische Methoden für Datenstruktur vom Typ `"lm"` — auf diese anzuwenden!
  - **print** wiederholt die Modellformel und gibt die Parameterschätzwerte aus
  - **summary** stellt zusammenfassend die Ergebnisse des Fits dar

- **plot** erstellt diagnostische Plots — siehe Abschnitt 7.1.1 (e)
- **coefficients** liefert die Parameterschätzwerte
- **effects** liefert die geschätzten Effekte bei einer ANOVA
- **fitted** . values liefert die gefitteten y–Werte
- **update** aktualisiert einen Modellfit — nach Hinzunahme / Weglassen eines Regressors oder einer Beobachtung
- **residuals** liefert die Residuen
- **deviance** liefert die Fehlerquadratsumme (RSS)
- **anova** erstellt die ANOVA–Tabelle der Ergebnisse; dessen Ergebnis kann auch "schön" **print** mit ausgegeben werden
- **lm. influence** identifiziert Hebelpunkte / einflussreiche Beobachtungen
- **predict** prognostiziert / liefert einen Schätzwert für einen "X–Wert", an dem kein "Y" vorliegt; siehe auch Abschnitt 7.1.1 (f)

- Hilfsfunktionen für Datenstruktur vom Typ `"lm"` — werden von

lm vewerdendet

- lm.fit führt gewöhnlichen Kleinste–Quadrate Fit durch
- lm.wfit führt gewichteten Kleinste–Quadrate Fit durch

R–Beispiel 7.1-4 [ein Regressionsbeispiel]:

```
library(MASS); data(hills}; attach(hills)
formul←time~climb+dist
erg←lm (formul)
print(erg); coefficients(erg)
summary(erg); deviance(erg)
s1←order(dist); s2←order(climb)
yf←fitted.values(erg)
par(mfrow=c(3,2)); plot(erg)
matplot(dist[s1],cbind(time,yf)[s1,],
        type="l",ylab="Zeit",xlab="Dist")
```


```
matplot(climb[s2],cbind(time,yf)[s2,],
        type="l",ylab="Zeit",xlab="Steig")
par(mfrow=c(1,1))
new<-data.frame(climb=3000,dist=20)
predict(erg,newdata=new,interval="confidence")
predict(erg,newdata=new,interval="prediction")
```

7.1.1 (e) Regressions–Diagnostik

- die `Summary`: — umfasst
  - Formel,
  - `fivenum`–Statistik der Residuen,
  - die Koeffizientenschätzungen mit jeweils einer Streuungsschätzung und der entsprechenden t–Statistik; letztere gibt an, mit welcher W–keit bei Vorliegen der 0 als tatsächlichem Parameter der Schätzwert für diesen Parameter

weiter von der 0 wegliegt, als die vorliegende Schätzung; d.h. kleine Werte deuten auf "signifikant von 0 verschieden", große Werte bedeuten, dass die Hypothese, der Parameter sei 0 nicht zu einem vernünftigen Niveau abgelehnt werden kann.

  - die Standardabweichung der Residuen mit Angabe der Freiheitsgrade — für eine weitere Verwendung in Tests
  - multiples r^2 und korrigiertes r^2 als Gütemaß für die Modellanpassung — siehe Abschnitt 7.1.1 (g)
  - F–Statistik als — für eine Verwendung in Tests auf Homoskedastizität, s.u.

- Residuenplot — eine der wichtigsten Diagnose–Methoden ist die Untersuchung der Residuen:
  - sind die Residuen symmetrisch verteilt? $\rightsquigarrow$ **boxplot**
  - sind die Residuen (annähernd) normalverteilt? $\rightsquigarrow$ **qqnorm**, c.f. Abschnitt 5.1.8
  - bei einem Plot (sofern möglich) $X \mapsto \mathrm{Residuum}(X)$: liegt

eine Struktur vor?

  - bei einem Plot Distanzmaß vom Zentrum gegen (Betrag der empirischen) Influenzkurve: (s.u.) liegen Hebelpunkte, einflußreiche Beobachtungen vor?

- die Hutmatrix
  - die Residuen sind nicht unabhängig; ihre Kovarianz ist $\mathbb{I} - H$ mit $H = X(X^\tau X)^{-1}X^\tau$ der *Hutmatrix*; — heißt so, weil sie den Beobachtungen Y einen "Hut" aufsetzt, indem $\hat{Y} = HY$ die gefittet Werte ergibt
  - hoher Diagonalwert $h_{i,i}$ von H deutet darauf hin, dass die i–te Beobachtung ein großer *Hebelpunkt* ist, wobei "groß" ungefähr 2 bis 3 mal p/n ist mit p der Zahl der Regressoren und n der Zahl der Beobachtungen
  - verschiedene Ideen, die Daten auf gleiche Skala zu bringen (vgl. Venables and Ripley (1999), p.157)
    * *standardisierte* Residuen: normiert durch $s\sqrt{1-h_{i,i}}$ mit

s^2 einer Skalenschätzung der Residuen

  * *studentisierte* (oder *Jackknife*) Residuen:
    sei $\tilde{y}_i$ der ohne Beobachtung i gefittete Wert für y_i; dann ist $\tilde{e}_i := \frac{y_i - \tilde{y}_i}{\tilde{s}_i}$ mit $\tilde{s}_i^2 = \text{(emp) Var}[y_i - \tilde{y}_i]$

- in R: Zugriff auf die (Diagonale der) Hutmatrix über **lm.influence** zugreifen

- Hebelpunkte
  sind Beobachtungen deren Faktor $(X^\tau X)^{-1} X^\tau$ sehr groß ist — ohne dass dabei das Residuum, also die empirische Modellabweichung an der Stelle groß sein muss, mit der Folge, dass sie einen großen Einfluß auf die Parameterschätzung haben

- **lm.influence**
  - wird angewendet auf ein Objekt vom Typ `"lm"`
  - der Rückgabewert besteht aus 3 Komponenten, `hat`, `coefficients` und `sigma`
  - `hat` die Diagonale von H

  - `coefficients`: eine Matrix, deren i-te Zeile aus $\theta_{\mathrm{LS}} - \tilde{\theta}_i$ besteht, wobei $\tilde{\theta}_i$ der Parameterschätzwert ist, den man ohne Beobachtung i erhält
  - `sigma` die $\tilde{s}_i$

- weitere Maße dfbeta, **dffits** , **covratio**, **cooks.distance**; siehe dazu Cook and Weisberg (1982), Belsley et al. (1980)
- Heteroskedastizität
  - die einzelnen Datenpunkte weisen unterschiedliche Varianz auf
  - fällt bei Residuenplot auf
  - kann man mit F–Statistik prüfen (nicht hier)

7.1.1 (f) Vorhersage

- Methode: wir verwenden als Prognose $\hat{y} := X\theta_{\mathrm{LS}}$
- problematisch falls X außerhalb der bisherigen Beobachtungen liegt (Extrapolation!)

- Unterscheide: Prädiktions– und Konfidenzintervall
  - ersteres berücksichtigt die zufällige Schwankung "ε" bei der vorherzusagenden Beobachtung — man würde ja nur eine Beobachtung an dieser Stelle X machen
  - letzeres gibt an womit man im Durchschnitt bei Beobachtung von X rechnen muß, wobei sich hier die "ε" 's schon herausgemittelt haben
- in R: durch **predict**, angewendet auf ein gefittetes Model von Typ `"lm"`
- Syntax:

```
predict (object, newdata, se.fit = FALSE, scale =
        NULL, df = Inf, interval = c("none",
        "confidence", "prediction"), level =
        0.95, type =c("response", "terms"),
        terms = NULL, ...)
```


- Argumente
  - `object`: Objekt vom Typ `"lm"`
  - `newdata`: Data–Frame (mit X–Werten), für die vorhergesagt werden soll
  - `se.fit`: (logisch) sollen Standardabweichungen mit angegeben werden?
  - `scale`: Skalenparameter zur Berechnung der Standardabweichungen
  - `df`: Freiheitsgrade für den Skalenparameter
  - `interval`: was für ein Intervall soll angegeben werden — Konfidenz– oder Prädiktionsintervall?
  - `level`: Toleranz / Konfidenz – Niveau
  - `type`: Art der Vorhersage — eine Beobachtung oder eine Modellgröße
  - `terms`: Falls `type="terms"`, welche Modellgrößen — default: alle

  - `...`: weitere Argumente, die an andere Methoden weitergereicht werden

- Rückgabewert
  - eine Matrix mit Spalten `fit` und evtl. `lwr`, `upr`, (falls `interval` spezifiziert ist); dabei sind `lwr` und `upr` untere und obere Intervallgrenzen des P.– oder K.–Intervalls
  - falls `se.fit TRUE` wird eine Liste zurückgegeben mit Komponenten
    * `fit`: wie gerade
    * `se.fit`: Standardabweichungen der Vorhersagen
    * `residual.scale`: Standardabweichung der Residuen
    * `df`: Freiheitsgrade für die Residuen

- Caveat: bei polynomialen Fits mit **poly** gibt es Probleme mit **predict**

- Ausweg: Verwendung von SafePrediction

7.1.1 (g) Modellwahl

- Idee bei geschachtelten Modellen:
  Ist beim Modellfit des höher–dim.-param. Modells ein Koeffizient nicht signifikant von 0 verschieden, so können wir die entsprechende Variable weglassen
- Signifikanz einer einzelnen Variable wird dabei mit dem t–Test geprüft
- Problem: obwohl ein Koeffizient nicht signifikant von 0 verschieden ist, kann er trotzdem signifikante Auswirkungen auf die anderen Koeffizienten haben
- $\rightsquigarrow$ Vergleich im F–Test: Fehlerquadratsumme mit und ohne den entsprechenden Regressor.
- r^2 und angepasstes r^2; r^2 ist dabei $1 - \texttt{RSS}/\texttt{TSS}$ mit `RSS` der Fehlerquadratsumme und `TSS` der Summe aller Beobachtungsquadrate; da höher–dim. parametrisierte Modelle

zu einem höheren r^2 führen, wird dies durch einen Strafterm im angepassten r^2 ausgeglichen.

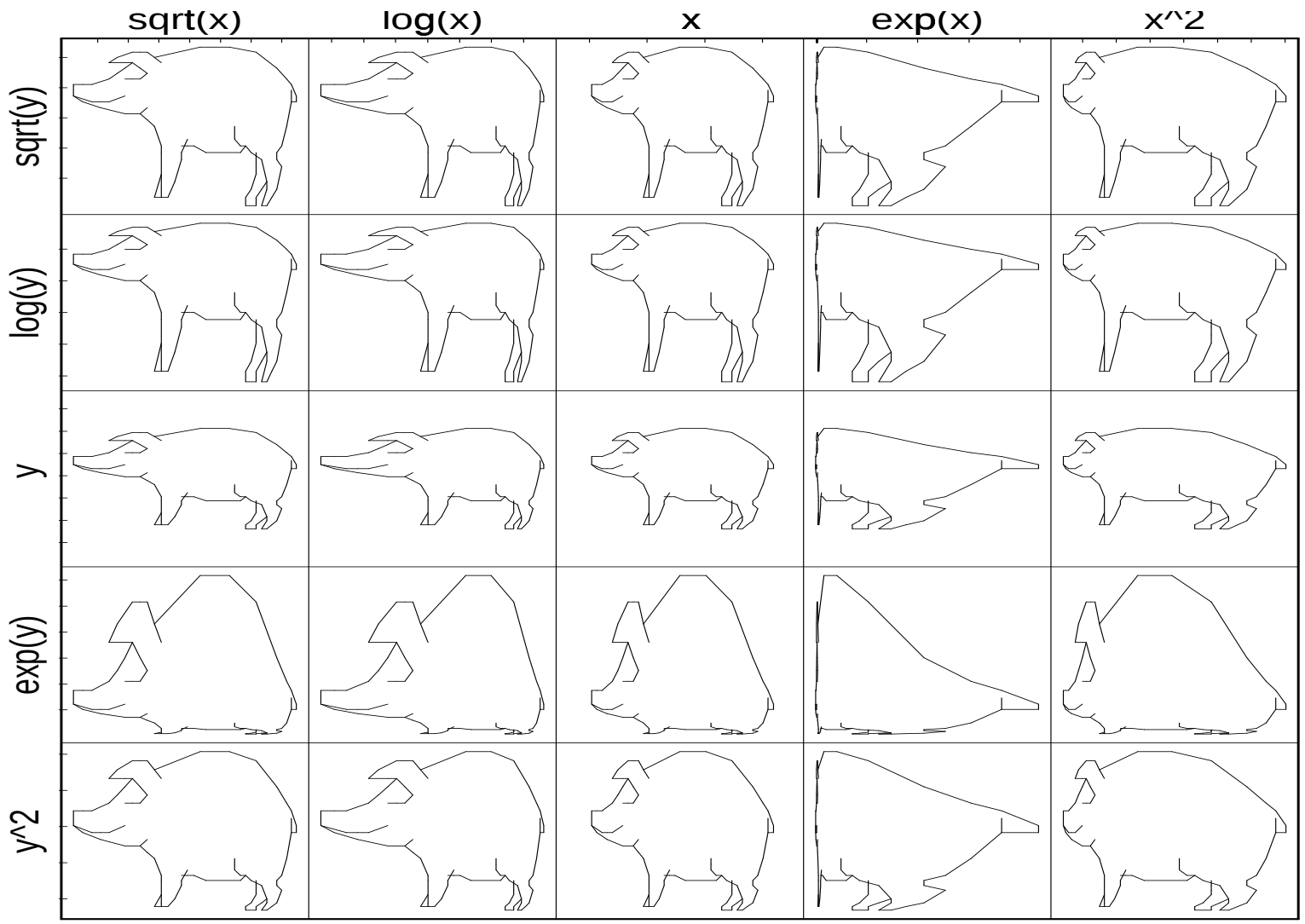
- ein anderer Ansatz ist Mallow's c_p, bei dem versucht wird, $\mathtt{RSS}/\hat{\sigma}^2 + 2p - n$ ungefähr auf p bekommen, wobei $\hat{\sigma}^2$ eine Schätzung für die Fehlervarianz auf Basis aller Beobachtungen
- AIC und BIC (*Akaike Information Criterium*, *Bayes Information Criterium*) legen die Loglikelihood der verschieden angepassten Modelle zugrunde; diese Kriterien gilt es zu minimieren
- in R realisiert durch leaps, **drop1**, **step**, **update**
- Caveat: Alle diese Verfahren sind mit Vorsicht zu genießen und können keine wissenschaftliche Argumentation ersetzen

R–Beispiel 7.1-5 [Modellanpassung]:

```
library(leaps)
data(state)
statedata←data.frame(state.x77,
 row.names=state.abb,check.names=T)
g←lm(Life.Exp ~ ., data=statedata)
x←model.matrix(g)[,-1]
y←statedata$Life
g←leaps(x,y)
plot(g$size,g$Cp)
abline(g$size,g$size)
```


7.1.1 (h) Wirkung von Datentransformationen

ein illustratives Beispiel aus Nagel et al. (1994), p. 183



7.1.1 (i) *Box-Cox*–Transformation

- Idee: in vielen Beispielen stabilisiert eine Transformation der Beobachtungen die Varianz und vermeidet so Heteroskedastizität
- ein systematischer Ansatz zum Auffinden einer "guten" Transformation ist der Box–Cox Ansatz: — vgl. Faraway (2002), Kapitel 8
- bei strikt positiven Beobachtungen wird versucht per Maximum Likelihood die Potenz λ der Datentransformation y^λ zu schätzen
- in R realisiert durch `boxcox`
- Syntax:

```
boxcox(object, lambda, plotit, interp,
       eps, xlab, ylab, ...)
```

- Argumente:
  - `object`: eine Formel oder ein Objekt vom Typ `lm` oder `aov`

  - `lambda`: ein λ–Gitter; default **seq**$(-2, 2,$**step**$=0.1)$
  - `plotit`: (logisch) soll das Resultat geplottet werden?
  - `interp`: (logisch) — soll Spline–Interpolation genutzt werden?
  - `eps`: Toleranz für `lambda = 0`; defaults 0.02
  - `xlab`, `ylab`: Achsenbschriftungen; per default `"lambda"` und `"log-Likelihood"`
  - `...`: die üblichen weiteren Parameter

- Rückgabewert — falls kein Plot angefertigt wird: ein Gitter $(\lambda_i, \log p_{\lambda_i})$
- beim Plotten wird zusätzlich ein 95% Konfidenzintervall um die ermittelte Maximalstelle $\hat{\lambda}$ angegeben
- falls `interp = TRUE` wird Spline–Interpolation zur Erzeugung eines glatteren Plots benutzt

7.1.2 Generalisiert Lineare Modelle

7.1.2 (a) ein einführendes Beispiel

- Modellierung der Kreditwürdigkeit eines Bankkunden
- Datensatz:
  - aus dem Datenarchiv der LMU München, unter Datensätze in Fahrmeir, Tutz (1996) $\rightsquigarrow$ "Kreditscoring zur Klassifikation von Kreditnehmern"
  - genaue Beschreibung der Daten siehe `http://www.stat.uni-muenchen.de/service/datenarchiv/kredit/kreditvar.html`
- Problem: Response binärwertig
- Ansatz: Nicht die Beobachtungen Y sind eine lineare Funktion der Regressoren, sondern ein Parameter der Verteilung der Y

- $\rightsquigarrow$ Transformation der Beobachtungen durch Link–Funktion η
- genauer $\mu = \mathrm{E}[Y]$ wird transformiert und ist ein $\mu = \eta^{-1}(X\theta)$
- Referenzen: Fahrmeir and Tutz (2001) und generell: McCullagh and Nelder (1989)

7.1.2 (b) wichtige Klassen

Verteilungsfamilien

- Binomial: Parameter $p \in [0, 1]$, zunächst noch nicht beliebig $\mathbb{R}$ wertig $\rightsquigarrow$ *Logit*, *Probit*
- Gamma: Parameter $\lambda \in [0, 1]$
- Normal: Parameter $\mu \in \mathbb{R}$
- Inverse-Gauss'sch: Parameter $\mu \in \mathbb{R}$
- Poisson: Parameter $\lambda \in [0, 1]$

Linkfunktionen

- bei binären Variablen
  - *Logit*–Trafo: $\log(p/(1-p))$
  - *Probit*–Trafo: $\Phi^{-1}(p)$
  - *cLog*–Trafo: $-\log(1-p)$
  - *cLoglog*–Trafo: $\log(-\log(1-p))$
- bei Zählvariablen
  - *Log*–Trafo: $\log(\lambda)$
- bei stetigen, positiven Variablen
  - *Log*–Trafo: $\log(\lambda)$
  - Inverse
  - Identität
  - Wurzel–Trafo: $\sqrt{p}$

Tabelle 7.1-6 [natürliche Linkfunktionen]:

Vertlg. Familie	kanonische Linkfunktion
Binomial	Logit
Gamma	Inverse
Gauß'sch	Identität
Invers-Gauß'sch	"1/mu^2"
Poisson	Log

7.1.2 (c) mögliche Aussagen / Fragestellungen

- Hat die Variable X einen Einfluß auf die Kreditwürdigkeit?
- Um wieviel erhöht sich das Risiko, dass der Kunde nicht zurückzahlt, wenn er in Kategorie x fällt?

7.1.2 (d) Modell–Einpassung in R

- der Befehl **glm**
- Syntax:

```
glm(formula, family = gaussian, data, weights =
    NULL, subset = NULL, na.action, start = NULL,
    offset = NULL, control = glm.control(...),
    model = TRUE, method = "glm.fit", x = FALSE,
    y = TRUE, contrasts = NULL, ...)
```

- Argumente
  - `family`: Modellfamilie — siehe Abschnitt zu Linkfunktionen
  - `weights`: (optionaler) Gewichtsvektor, der beim Einpassen benutzt wird
  - `start`: Parameterstartwerte für das iterative Schätzverfahren
  - `etastart`: Startwert für η
  - `mustart`: Startwert für μ

- `control`: eine Liste mit Steuerungs/Kontrollgrößen für das iterative Schätzverfahren, wie Abbruch–ε, maximale Iterationszahl etc.
- `method`: Schätzmethode — z.Z. nur `glm.fit` implementiert, das iterativ gewichtete Kleinste Quadrate (IWLS) benutzt
- `formula`, `data`, `subset`, `na.action`, `offset`, `model`, `x`, `y`, `contrasts`, `intercept`, `...`: wie bei der gewöhnlichen linearen Regression

- Rückgabewert ein Objekt vom Typ `"glm"` mit Elementen wie `"lm"`

- zusätzlich:
  - `linear.predictors`: die lineare Einpassung im Raum der Linkfunktion
  - `deviance`: bis auf eine Konstante -2 mal die maximierte Log-likelihood.
  - `aic`: AIC–Wert zur Bestimmung der Modellordnung
  - `null.deviance`: Eichwert des AIC an einem Null–Modell, zu vergleichen mit der `deviance`
  - `iter`: benötigte Zahl an Iterationen beim Fit
  - `weights`: die zuletzt benutzten the Gewichte
  - `prior.weights`: Start–Gewichte sofern angegeben
  - `df.residual`: Freiheitsgrade der Residuen
  - `df.null`: Freiheitsgrade der Residuen im Nullmodell.
  - `converged`: (logisch) Hat das Verfahren die gewünschte Genauigkeit geliefert?
  - `boundary`: (logisch) liegt der Parameter–Fit auf dem Rand

des Definitionsbereichs?

  - `family`, `control`: Parameter aus der Eingabe
  - `xlevels`: (sofern relevant) eine Aufstellung aller Faktor–Niveaus, die zum Fitten benutzt worden sind

- Caveat: Bei Zähl–/ Binomialmodellen muß man als Y–Vektor die absoluten Treffer–/ Fallzahlen angeben!

7.1.3 ein Beispiel

R–Beispiel 7.1-7 [das Kreditbeispiel vom Anfang]:

```
kredit ← data.frame(read.table(file=
    "C:/programme/R/uebung/kredit.txt", header=T))
attach(kredit)
formul←kredit~laufzeit+hoehe+sparkont+rate

erg←glm(formula=formul,data=kredit,
```


```
            binomial(link = "probit"))
summary(erg)
par(mfrow=c(2,2))
plot(erg)
par(mfrow=c(1,1))
```

7.1.3 (a) Modellwahl — die Deviance

- anders als bei klassischen linearen Modellen sind hier nur asymptotische Aussagen möglich;
- wichtiges Kriterium ist die *Deviance* (s.o.)
- in R steht dazu die generische Funktion **deviance** zur Verfügung

7.2 Elemente Multivariater Statistik

7.2.1 die multivariate Normalverteilung

- Erzeugung einer Verteilung $\mathcal{N}_p(\mu, \Sigma)$ siehe Übung
- multivariate Normalverteilungstests: Idee
  $X \sim \mathcal{N}_p(\mu, \Sigma) \quad \Longleftrightarrow \quad a^\tau X \sim \mathcal{N}_1(\mu_a, \sigma_a^2)$ für alle $a \in \mathbb{R}^p$
- siehe auch Pakete mvtnorm, ellipse

7.2.2 graphische Methoden

- siehe Kapitel 4

7.2.3 Allgemeines

7.2.3 (a) generelle Fragestellungen

- Dimensionsreduktion
- Klassifikation mit / ohne Training
- Aufdecken interessanter Strukturen

- Wie kann man die Daten gut in einem niedrig–dimensionalen Raum repräsentieren?
- Mit welchen Variablen kann man gut klassifizieren?
- Gibt es gut unterscheidbare Teilklassen und wenn ja wieviele?

7.2.3 (b) allgemeine Pakete zur mult.var. Statistik

- mva, modreg, MASS, multiv

7.2.3 (c) allgemeine Literatur zur mult.var. Statistik

- Einstieg: Flury and Riedwyl (1983), Flury (1997), Härdle, W. and Simar, L. (2003)
- weiterführend: Anderson (1984), Mardia et al. (1979)

7.2.4 Hauptkomponenten– und Faktoranalyse

- Fragestellungen
  1. In welchen Richtungen variieren die Daten vorzugsweise?
  2. Gibt es "neue" Variablen, die aus den alten als Linearkombinationen hervorgehen, und die Daten besser beschreiben als die alten?
  3. "Gute" zweidimensionale Abbildung einer hochdimensionalen Punktewolke
- Herangehensweise / Idee:
  - Spektralzerlegung (SVD) der (empirischen) Kovarianz–/Korrelationsmatrix der Daten
  - Projektion auf die ersten (größten) Eigenräume
  - bei der Faktoranalyse: nur noch verwenden der ersten (größten) Eigenräume — als latente Variablen

- Implementation in R — alles in Bibliothek mva
  - princomp, prcomp: Hauptkomponentenanalyse
  - factanal , varimax: Faktoranalyse
  - **summary** (generische Methode)
  - spezielle Plots: **plot** (generische Methode): erzeugt *screeplot*, biplot : siehe Kapitel 4

7.2.5 Multidimensional Scaling

- Fragestellung: aus einer Abstandsmatrix erzeuge eine niedrig (2)–dimensionale Konfiguration, so dass die Abstände der Punkte in der Konfiguration möglichst genau denen in der Abstandsmatrix entsprechen
- Verschiedene Abstandsbegriffe, $\rightsquigarrow$ *similarities*, *dissimilarities*, *Ultrametriken*
- numerische, iterative Algorithmen
- Implementation in R
  - in mva: cmdscale — klassisches (euklidisches) MDSCAL
  - dazu (ebenfall in mva): dist zur Erzeugung einer Abstandsmatrix
  - in MASS: isoMDS, sammon — zwei nicht–metrische MDSCAL's von Venables and Ripley (1999)

7.2.6 Cluster–Analyse

- Fragestellung: finde eine Partition der Stichprobe in homogene Teile;
- vorgegebene Zahl an Clustern $\leftrightarrow$ mit zu bestimmen
- Strategien
  - hierarchisches Clustern (agglomerativ, verteilend)
    - single / complete / average linkage
    - weitere: Ward, centroid, median, mcquitty
  - k–Mittel
  - modellbasiertes Clustern
    - Modell: habe p unterschiedliche Stichproben mit vollständig/bis auf endl.-dim. Parameter bekannte Verteilung; diese wurden vermischt;
    - Aufgabe: rekonstruiere die Mixing–Indikatoren

- Implementation in R
  - hierarchisches Clustern — in mva
    - hclust — erzeugt Aufteilungshierarchie als Objekt vom Typ `hclust`; benötigt eine Abstandsmatrix (erzeugt durch dist)
    - dazu **plot** (oder synonym plclust) — Ausgabe des Hierarchie–Baums (*Dendrogramm*), **identify** . hclust
    - cutree zerlegt eine Hierarchie in vorgegebene Zahl an Cluster / oder längs vorgegebener Hierarchie–Ebene
  - k–Mittel — in mva: kmeans
  - modellbasiertes Clustern: Paket mclust; in `S-Plus`: mclust, mclass, mreloc
  - Extra–Pakete: cluster , cclust

7.2.7 Diskriminanzanalyse

- Fragestellung: Gegeben eine Lern–Stichprobe mit bekannter Aufteilung in Untergruppen, finde heraus, wie man bei einer neuen Beobachtung gut vorhersagen kann, zu welcher Gruppe sie gehört
- Methoden
  - Diskriminanzanalyse nach Fisher: versucht, den Quotienten aus Intra– und Inter–Gruppenvarianz zu maximieren
  - lineare Diskriminanzanalyse (Hypothese: alle Gruppenkovarianzen gleich)
  - bei Normalität: quadratische Diskriminanzanalyse
- Implementation in R
  - lineare Diskriminanzanalyse in MASS: lda
  - quadratische Diskriminanzanalyse in MASS: qda
  - Extra–Paket: mda

7.3 Zeitreihenanalyse

7.3.1 Einführung

7.3.1 (a) Grundlagen

- im Unterschied zum $\mathrm{u.i.v.}$–Setup nun abhängige Beobachtungen
- $\Rightarrow$ Reihenfolge / Anordnung der Beobachtungen wichtig
- Bezeichnung: das Objekt $\{X_t(\cdot)\}_t$ heißt *stochastischer Prozess*, eine Realisation $\{X_t(\omega)\}_t$ heißt Zeitreihe
- um dennoch ähnliche Schlüsse wie im $\mathrm{u.i.v.}$–Setup ziehen zu können, sind folgende Begriffe nötig
  - *Stationarität*: Hätte die beobachtete Zeitreihe von der Verteilung her betrachtet auch an einem anderen Startzeitpunkt starten können?

– um mit nur einer Realisation (einer Zeitreihe) Inferenz auf weitere Beobachtungen möglich zu machen: Vertauschbarkeit der Mittelung "über ω" mit der "über t" $\rightsquigarrow$ *Ergodizität*

7.3.1 (b) Literatur

- Einstieg: Schlittgen and Streitberg (1987), Brockwell and Davis (2002), Wei (1990)
- weiterführend: Hamilton (1994), Brockwell and Davis (1991), Harvey (1993), Shumway and Stoffer (2000), Durbin and Koopman (2001)
- spezielle Themen: Granger and Newbold (1986), Gouriéroux (1997), Rothman (1999)

7.3.1 (c) wichtige allgemeine Pakete in R

— zu beziehen auf CRAN

- in der üblichen Distribution mit dabei: **ts**
- allgemein: tseries von Adrian Trapletti, gepflegt von Kurt Hornik,

- gerade für multiple Zeitreihen und Zustandsraummodelle: dse, von P. Gilbert
- Signalerkennung: Rwave, in S von Rene Carmona, in R portiert und gepflegt von Brandon Whitcher
- Erkennen von Strukturbrüchen strucchange, von Achim Zeileis, Friedrich Leisch, Bruce Hansen, Kurt Hornik, Christian Kleiber, Andrea Peters, gepflegt von Achim Zeileis
- ökonometrische Modelle: sem, von John Fox
- periodische Zeitreihenmodelle: pear, in S von A. I. McLeod, in R portiert und gepflegt von Mehmet Balcilar
- nichtparametrische Statistiken für Zeitreihenmodelle: pastecs, von Frederic Ibanez, Philippe Grosjean und Michele Etienne, gepflegt von Philippe Grosjean
- Hilfspaket zum Umgang mit Zeiteinheiten: chron Original in S von David James, in R portiert und gepflegt von Kurt Hornik.

7.3.1 (d) spezielle Klassen/Befehle in R zum Umgang mit Zeitpunkten

- neben Beobachtungsindex ein weitere Index: der Beobachtungszeitpunkt ("Time Stamp")
- hilfreich zum Konvertieren von verschiedenen Datums-/Zeitformaten: Paket `chron` — siehe **require**(chron); ?chron; ?times; ?dates,
- bei regulärem Gitter ($t_i - t_{i+1} = \text{const}$) einfache Umrechnung; umgesetzt in (S3)-Klasse `ts` (aus `stats` Paket)
- sonst: schwieriger; dazu
  - Attribut-Ansatz `tframe` aus Paket `tframe`: Beobachtungszeitpunkte werden kein Teil der Daten sondern ein Attribut des Datenobjekts
  - (S4)-Klasse `its` aus Paket `its`: Beobachtungszeitpunkte der Klasse `POSIXct`

  - (S3)-Klasse `irts` aus Paket `tseries`: Beobachtungszeitpunkte der Klasse `POSIXct`
  - (S3)-Klasse `zoo` aus Paket `zoo` —≙̂Z.'s [=Achim Zeileis'] ordered objects; hier müssen die Beobachtungszeitpunkte nur irgendwelche ordinale Merkmale sein!

- als (S3)/(S4)-Objekte handelt es sich hierbei jeweils um Datenstrukturen, bei denen eine Reihen von (generischen) Funktionen spezielle Funktionalität erhalten (s.u.)

Konstruktoren: die Funktionen **ts**(), its(), irts(), tframe() und zoo()

- **ts**() erzeugt ein uni– oder multivariates Zeitreihen–Objekt der Klasse `ts` aus Argumenten:
  - `data`: die Daten;
  - `start`: Beobachtungstartpunkt
  - `end`: Beobachtungendpunkt
  - `frequency`: Beobachtungen/Zeiteinheit

  - `deltat`: Zeitabstand zwischen zwei Beobachtungen
  - `ts.eps`: ab wann gelten zwei Zeitpunkte gleich?
  - `class`: `mts` (multiple Zeitreihe); `ts` (univariate Zeitreihe) oder `NULL` sonst
  - `names`: Namen der Dimensionen der Zeitreihe
- `zoo()` erzeugt ein Objekt der Klasse `zoo` aus Argumenten:
  - `x`: die Daten;
  - `order.by`: Index —"Beobachtungzeitpunkte"— anhand deren man die Beobachtungen anordnen kann
  - siehe auch `zoo-quickref.pdf` bzw. `zoo.pdf` über HTML-Hilfe zum Paket und dort `overview`
- its (), irts (), tframe() erzeugen jeweils Objekte der Klassen its , irts , tframe — siehe deren spezielle Hilfe

- überladene Methoden
  - Indexoperationen `[....]`
  - Typumwandlungen `as.ts`, `is.ts`, `as.zoo`, `is.zoo`, `as.its`, `is.its`, `as.irts`, `is.irts`
  - Ausgaben `plot`, `lines`, `print`
  - Verschmelzen von Zeitreihen (mit unterschiedlichen Beobachtungszeitpunkten) **cbind** (= **cbind.ts**), **ts.union**, **ts.intersect** bzw. allgemeiner und leistungsfähiger **merge** für `zoo` Objekte
  - **na.omit.ts**: bei Missings wird `omit` nur an den Enden zugelassen
  - übliche Arithmetikoperationen

- spezielle Methoden
  - **aggregate** berechnet Zusammenfassungen über disjunkten Zeitintervallen
  - **diff** erzeugt die Zeitreihe $Y_t := X_t - X_{t-i}$
  - **end** Zeitpunkt der letzten Beobachtung
  - **frequency** Zahl der Beobachtungen pro Zeiteinheit
  - **deltat** Zeitabstand zwischen zwei Beobachtungen (`ts`)
  - **time** gibt Beobachtungszeitpunkte zurück / ändert sie
  - **cycle** erzeugt eine Zeitreihe durch Vorgabe der Beobachtungszeitpunkte innerhalb eines Zykluses
  - **start** Zeitpunkt der ersten Beobachtung
  - **tsp**, **tsp**$\leftarrow$ Ausgabe/Modifikation der Zeitreihenattribute
  - **window** Herausfiltern einer Teilzeitreihe innerhalb eines Fensters

- spezielle **apply**-artige Funktionen für Auswertungen auf gleitenden Fenstern (bei `zoo`)
  - rapply — wie **apply** auf Fenster der Länge `width`
  - speziell: rollmean, rollmax, rollmedian
- Interpolation/Fortschreibung —(Paket `zoo`)
  - `na.approx` ersetzt missings durch lineare Approximation der umliegenenden beobachteten Werte
  - `na.locf` ersetzt missings durch letzten beobachteten Wert
- weitere Methoden aus Paket `ts`
  - lag — mit lag.**plot** (auch in `zoo`)
  - **ts**.**plot**, month.**plot**

7.3.2 Autokovarianz und Spektrum

7.3.2 (a) Begriffe

- wesentliches Instrument zur Beschreibung von Abhängigkeiten: die Kovarianz $g^X(t,s) = \mathrm{Cov}(X_t, X_s)$ — die *Autokovarianz*
- im stationären Fall $g^X(t,s) = \gamma^X(|t-s|)$; $\gamma^X : \mathbb{N}_0 \to \mathbb{R}$ heißt Autokovarianzfunktion oder ACF von X
- Satz aus der Fourier–Theorie (Herglotz): *Autokovarianzfunktionen sind genau die Fouriertransformierten von beschränkten Maßen auf* $(-\pi;\pi]$, *und diese Relation ist* $1:1$
- das γ^X entsprechende Maß heißt *Spektralmaß*; falls das *Spektralmaß* eine (Spektral–)Dichte besitzt, erhält man diese als

$$f(\lambda) := \frac{1}{2\pi} \sum_{n\in\mathbb{Z}} \gamma^X(n) \exp(-in\lambda) \qquad \text{für } \lambda \in [-\pi,\pi] \quad (7.3.1)$$

- Idee: Stelle die Zeitreihe als eine Überlagerung von *Frequenzen* dar $\rightsquigarrow$ Argumentation im Frequenzbereich $\leftrightarrow$ Argumentation im Zeitbereich

7.3.2 (b) Realisation in R

- Autokovarianz — aus Paket **ts**
  - acf berechnet die (empirische) Autokovarianz /–korrelation
  - pacf berechnet die (empirische) *partielle* Autokovarianz /–korrelation
  - zu den letzten beiden: spezielle **plot**–Funktion
- Spektraldichte /–verteilung — aus Paketen **ts** / MASS
  - ähnliche Probleme wie bei der Dichteschätzung
  - Methoden:
    * Bestimmung aus Schätzung der ACF
    * Bestimmung mit FFT
    * Tapering

- in R — in **ts**
  - Bestimmung aus Schätzung der ACF: spec.ar
  - Bestimmung mit FFT: spec.pgram
  - Tapering: spec.taper
  - weitere Glättung durch gleitende Mittel mit `span`
- alle in einer "Hülle" aufrufbar: spectrum
- dazu spezielle **plot**–Methode:
  - Angabe eines 95%–Konfidenzintervalls
  - Darstellung der Breite des Glättungsfensters
- auch: empirische Verteilungsfunktion des Spektralmaßes als kumulatives Periodogramm — in MASS — cpgram

7.3.3 ARIMA–Modelle

7.3.3 (a) Modelldefinition

- Modellgleichung für ARMA

$$\sum_{i=0}^{p} \phi_i X_{t-i} = \sum_{j=0}^{q} \xi_j V_{t-j} \tag{7.3.2}$$

  mit Beobachtungen X_t (+ Anfangswerten) und Innovationen V_i

- Backshift:
  setzen wir $BX_t := X_{t-1}$ und
  $\Phi(z) := \sum_{i=0}^{p} \phi_i z^i$, $\Xi(z) := \sum_{j=0}^{q} \xi_i z^i$, so wird aus (7.3.2)

$$\Phi(B)X_t = \Xi(B)V_t \tag{7.3.3}$$

- Identifizierbar falls Φ und Ξ keine gemeinsame Nullstellen
- Stationarität und Invertierbarkeit $\iff$ alle Nullstellen z über $\mathbb{C}$ von Φ bzw. Ξ sind $|z| > 1$

- "Stationarisierung" durch Differenzenbildung:
  mit $\Delta(B) = 1 - B$ heißt das Modell

$$\Delta^d \Phi(B) X_t = \Xi(B) V_t \qquad (7.3.4)$$

  ARIMA–Modell der Ordnung (p, q, d)

7.3.3 (b) Realisation in R

- Simulation: arima.sim
- Parameterschätzung: arima, ar
- Vorhersage: **predict** .Arima
- Bestimmung der Modellordnung — zB mit arima, tsdiag

7.3.4 Trend– und Saison–Bereinigung

- Zerlegungsmodelle
  - (additive) Zerlegung in Trend, Saison und unstrukturiertes Rauschen —in R: StructTS

– saisonale ARIMA–Modelle —auch mit arima

7.3.5 Multiple Zeitreihen

7.3.5 (a) Einführung

- in vielen Situationen: Beobachte zu einem Zeitpunkt mehrere Phänomene simultan $\rightsquigarrow$ vektorwertige oder *multiple* Zeitreihen
- entsprechend Autokovarianz– bzw. Autokorrelationsfunktion matrixwertig $\rightsquigarrow$ Kreuzkorrelation / –spektrum
- auch "Stationarisierung" durch Bezug auf eine (nicht stationäre) Referenzzeitreihe $\rightsquigarrow$ Kointegration
- Modellierung komplexer Wechselwirkungen durch *Transfer–Functions*, c.f. Hamilton (1994)
- entsprechend vektorwertige ARIMA–Modelle $\rightsquigarrow$ VARIMA

7.3.5 (b) Realisation in R

—in dse

- Simulation: simulate
- Modelldefinition: ARMA (ARMA), SS (Zustandsraummodell)
- Parameterschätzung: est . VARX.ar, est . black . **box**
- Bestimmung der Modellordnung est . black . **box**, reduction . Mittnik
- Vorhersage forecast , horizons . forecast

7.3.6 Zustandsraummodelle

7.3.6 (a) Modelldefinition

- Modell in zwei Ebenen: Beobachtungs– und Zustandsebene
- im linearen, zeitdiskreten, Euklidischen Fall:

$$\beta_t = F_t\beta_{t-1} + v_t \quad (7.3.5)$$

$$y_t = Z_t\beta_t + \varepsilon_t \quad (7.3.6)$$

- interessierende Größe $\{\beta_t\}$, Beobachtungen: y_t
- je nach Horizont der zur Verfügung stehenden Beobachtungen Glättungs–, Filter– oder Vorhersageproblem
- sehr flexibel, extrem weiter Anwendungsbereich

7.3.6 (b) der Kalman–Filter

- mithilfe der Methode der kleinsten Qaudrate: rekursives Verfahren zur Schätzung der $\{\beta_t\}$: der *Kalman–Filter*
- im linearen, zeitdiskreten, Euklidischen Fall:

$$\beta_{0|0} = a_0 = \mathrm{E}[\beta_0] \tag{7.3.7}$$

$$\beta_{t|t-1} = F_t \beta_{t-1|t-1} \tag{7.3.8}$$

$$\beta_{t|t} = \beta_{t|t-1} + M_t(y_t - Z_t \beta_{t|t-1}) \tag{7.3.9}$$

- M_t heißt Kalman–Gain
- analoge Rekursionen für das Glättungs– und Prognoseproblem

7.3.6 (c) Realisation in R

- in dse siehe Abschnitt 7.3.5 (b)

- in robKalman — eine Bayreuth/Wiener Entwicklung und noch im "α-Stadium"...— verfügbar unter

  http://www.uni-bayreuth.de/departments/math/org/mathe7/robKalman:

  - Paket für robustes Kalman-Filtern; siehe **require**(robKalman);?robKalman
  - (vorläufig) noch keine Verwendung von (S3)/(S4) Klassen
  - Simulation mit simulateState, simulateObs (inklusive Kontamination, wenn gewünscht)
  - Filtern mit KalmanFilter, bzw. robuste Alternativen: rLSFilter (auch multivariat), ACMfilter (univariat)

- in sspir :
  - Definition des Zustandsraums mit SS
  - sehr flexibel: auch GLM-Bestandteile und nichtlineare Übergänge möglich —siehe ?ssm
  - Simulation mit simulate
  - Kalman Filter kfilter

– Kalman Glätter smoother

R–Beispiel 7.3-1 [Kalman-Filter]:

```
data(mumpsdat)
time <- 1:nrow(mumpsdat)
m3 <- ssm( mumps ~ -1 + tvar(polytime(time,1)) +
           tvar(polytrig(time,12,1)),
           family=poisson(link=log),time=time,
           data=mumpsdat)
### ssm extrahiert SS Modell aus dieser Formel
m3$ss$phi["epsilon"] <- 0
m3$ss$phi["polytime(time, 1)time0"] <- 0
m3$ss$phi["polytime(time, 1)time1"] <- 0.0005
m3$ss$phi["polytrig(time,12,1)"] <- 0.0001
diag(m3$ss$C0) <- 1
m3.fit <- kfs(m3)
```


7.3.7 (G)ARCH–Modelle

7.3.7 (a) Modelldefinition

- ein nichtlineares Modell: Variabilität heute ist Funktion hängt ab von der Größe der Beobachtung gestern (und der Variabilität gestern im Fall von GARCH)
- Modell

$$X_t = s_t v_t \tag{7.3.10}$$

$$s_t^2 = 1 + \sum_{i=1}^{p} \alpha_i X_{t-i}^2 + \sum_{j=1}^{q} \beta_j s_{t-j}^2 \tag{7.3.11}$$

- Anwendung im Finanzbereich:
  Idee: Variabilität der Kurse hängt ab vom Marktvolumen

- Simulation — händisch, siehe Beispiel zu ?garch
- Parameterschätzung garch
- Bestimmung der Modellordnung zB. garch, **summary**(auf ein `garch`-Objekt angewandt)
- Vorhersage **predict**

7.3.8 weitere finanzmathematische Modelle

- z.B. CAPM, Sharpe ratio, Maxloss:
  portfolio .**optim**, sharpe, maxdrawdown
- übrigens auch: **get**. **hist** .**quote**, um historische Kurse von `Yahoo!Finance` herunterladen
- siehe Hilfe zu tseries

7.3.9 Tests aus Paket `tseries`

7.3.9 (a) U.i.v.-Hypothese

- BDS-Test `bds.test`

7.3.9 (b) Normalität

- Jarque-Bera-Test `jarque.bera.test`

7.3.9 (c) Zufälligkeit bei binären Merkmalen

- Run-Test `runs.test`

7.3.9 (d) Einheitswurzeltests

- Augmented Dickey-Fuller-Test `adf.test`
- Phillips-Perron-Test `pp.test`
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) `kpss.test`

7.3.9 (e) Kointegrationstests

- Phillips-Ouliaris-Test `po.test`

7.4 Geostatistik

7.4.1 Grundlagen

7.4.1 (a) Situation

- habe Messungen Y, die abhängen
  - vom Ort der Messung
  - evtl. Messzeitpunkt
  - Zufall
- also $Y = Y_{x,t}(\omega)$, x der Ort, t die Zeit
- Beispiele: Modellierung von Wetter, Metalleinlagerungen, Gewässerverunreinigung
- Schwierigkeit: niedrigdimensionale Modellierung der Abhängigkeit

7.4.1 (b) Begriffe der räumlichen Statistik

- entsprechende Umsetzung der Begriffe *Stationarität*, *Ergodizität*
- Analogon zur ACF: *Variogramm*, *Korrelogramm*
- in R, genauer in spatial : Funktionen correlogram, variogram mit Modellierung `expcov`, `gaucov`, `sphercov`
- Caveat: *Nugget–Effekt* für Kovarianzen zwischen Punkten, die sehr nah beieinander liegen

7.4.1 (c) Literatur

- anwendungsorientiert:
  Davis (1986), Isaaks and Srivastava (1989)
- Einstieg (allgemein):
  Unwin (1981), Cliff and Ord (1981), Ripley (1981),
- weiterführend: Cressie (1991), Ripley (1991)
- spezielle Themen: Stoyan et al. (1995)

- (zum Teil kommentierte) Literaturlisten:
  - http://www.geo.sbg.ac.at/staff/lorup/lv/geostats/literatur_kommentiert.htm
  - http://www.geo.sbg.ac.at/staff/lorup/lv/geostats2000/literatur_kommentiert.htm
  - http://slc.mathematik.uni-ulm.de/cgi-bin/vorlinfo.pl?lid=MAS020&semester=SS2002
- Links im WWW:
  - http://www.geog.fu-berlin.de/j̃krywkow/harald/geostatistik/referat.html
  - http://www.geocities.com/Tokyo/Flats/7335/medical_geography.htm
  - http://www.geo.sbg.ac.at/staff/lorup/lv/geostats2000s/links.htm
  - http://www.spatial-statistics.com/spatial_links_index.htm
  - http://www.statistical.org/

7.4.1 (d) wichtige allgemeine Pakete in R

— zu beziehen auf CRAN

- der üblichen Distribution mit dabei: spatial ,
- allgemeine Bibliotheken:

  - geoR von Paulo J. Ribeiro Jr Peter J. Diggle gepflegt von Paulo J. Ribeiro Jr
  - geoRglm — GLM's für Random Fields, von Ole F. Christensen und Paulo J. Ribeiro Jr gepflegt von Ole F. Christensen
  - in BT naheliegend — auch für Extremwertstatistik: RandomFields von Martin Schlather
  - ein objekt–orientierter Rahmen für geostatistische Modellierung: sgeostat, Orignal in S von James J. Majure; portiert in R, erweitert und gepflegt von Albrecht Gebhardt
- Raumzeitprozesse:
  - pastecs — siehe Abschnitt 7.3.1 (b),
  - spatstat von Adrian Baddeley und Rolf Turner, gepflegt von Adrian Baddeley
- Pakete von Roger Bivand
  - Räumliche Abhängigkeit — Gewichtsschemen, Modelle und Statistiken: spdep, von Roger Bivand, mit Beiträgen von

Nicholas Lewin-Koh und Michael Tiefelsdorf, gepflegt von Roger Bivand

  - Räumliche und raum–zeitliche Analyse von Punktprozessen: splancs von Barry Rowlingson und Peter Diggle, adaptiert, in ein R–Paket "geschnürt" und gepflegt von Roger Bivand, pcp–Funktionen von Giovanni Petris

- Hilfspakete zur Triangulierung:
  - Delaunay Triangulierung und Dirichlet / (Voronoi) Tesselation (Pflasterung?): deldir , von Rolf Turner
  - Triangulierung auf einem ungleichmäßigen, 2–dimensionalen Gitter beobachteter Daten: tripack , Fortran Code von R. J. Renka, in R portiert und gepflegt von Albrecht Gebhardt mit Beiträgen von Stephen Eglen und Sergei Zuyev

7.4.2 Interpolation und Kriging

- Problem: Habe Messungen nur auf Gitter $\rightsquigarrow$ Schätzung an Nichtgitterpunkten
- Methoden:
  - niedrigdimensionale, polynomiale Interpolation
  - lokale Trendflächen
  - Ausnutzung der Kovariogramm–Struktur: *Kriging* — Vorhersage eines linearen Modells $Y(x) = f(x)^\tau \beta + \varepsilon(x)$, indem man die Prognose $\hat{\varepsilon}(x)$ nicht auf 0 setzt, sondern auf die aus den Residuen hervorgehende Prognose für $\varepsilon(x)$
- in R — in Paket spatial
  - polynomiale Interpolation: surf.ls zur Bestimmung des Polynoms und trmat zur Auswertung auf (neuem) Gitter
  - lokale Trendflächen — in Paket modreg: loess zur Bestimmung der lokalen Fläche und **predict** zur Auswertung

auf (neuem) Gitter

– Kriging: anstelle von trmat nun prmat zur Auswertung der Kriging–Vorhersage, semat zur Auswertung der Vorhersage des Residuums auf (neuem) Gitter

7.4.3 Punktprozesse

- modelliert das (zufällige) Auftreten von Phänomenen auf einer "Beobachtungsfläche" (–region, –menge)
- Beispiel: Verbrechen an einem bestimmten Ort, Ausbruch einer Krankheit
- Standardprozess: *Poisson–Prozess* — keine "Gleichzeitigkeit"
- Begriffe:
  - *Intensität* λ in Abhängigkeit einer betrachteten Teilmenge: erwartete Zahl an Ereignissen in dieser Menge
  - die K– bzw. L–Funktion:
    $\lambda K(t) := \mathrm{E}\,\#\{\text{Punkte innerhalb eines Abstands } t\}$,

$L(t) := \sqrt{K(t)/\pi}$;
Schätzung durch Kfn; durchschnittliches K mit Kaver, maximales/minimales K mit Kenvl

– Einlesen der Daten und setzen des interessierenden Gebiets mit ppinit, neu setzen dann durch ppregion

- Alternative zum Poissonprozess: der *Strauss–Prozess* in spatial realisiert durch Strauss (für Simulationen) und pplik zum Einpassen der Parameter an reale Daten

R–Beispiel 7.4-1 [Piniendatensatz aus Schweden]:

```
library(spatial)
# Einlesen: Daten der Pinien + Def.bereichs
pines <- ppinit("pines.dat")
par(mfrow=c(2,2), pty="s")
```


```
plot(pines, xlim=c(0,10), ylim=c(0,10),
     xlab="", ylab="",xaxs="i", yaxs="i")
plot(Kfn(pines,5), type="s",
     xlab="distance", ylab="L(t)")
lims ← Kenvl(5, 100, Psim(72))
lines(lims$x, lims$l, lty=2)
lines(lims$x, lims$u, lty=2)
ppregion(pines)
plot(Kfn(pines,1.5), type="s",
     xlab="distance", ylab="L(t)")
lims ← Kenvl(1.5,100,Strauss(72, 0.2, 0.7))
lines(lims$x, lims$a, lty=2)
lines(lims$x, lims$l, lty=2)
lines(lims$x, lims$u, lty=2)
pplik(pines, 0.7)
lines(Kaver(1.5,100,Strauss(72, 0.15, 0.7)),lty=3)
```


8 fortgeschrittene Programmierung

8.1 R als objektorientierte Sprache

8.1.1 Paradigmen objektorientierter Programmierung (OOP)

8.1.1 (a) Literatur

- Booch (1995)
- Stroustrup (1987)

8.1.1 (b) allgemeine Prinzipien in der Programmierung

- prozedurales Programmieren:

  - Grundsituation: einzelner Programmierer
  - $\rightsquigarrow$ aus Übersichtlichkeitsgründen: Aufteilen der Programmierung in einzelne Funktionen/Prozeduren
  - Paradigma: *"Entscheide, welche Prozeduren Du willst; verwende den besten Algorithmus, den Du kennst."*

- modulares Programmieren
  - Grundsituation: Gruppe von Programmierern
  - $\rightsquigarrow$ Aufteilen der Programmierung in einzelne Module von unterschiedlichen Autoren
  - *Modul* $\hat{=}$ Menge von Prozeduren + Daten die diese manipulieren
  - Paradigma: *"Entscheide, welche Module Du brauchst; zerlege die Daten so in Module, dass die Daten in Module eingekapselt sind."*
  - Transfer von Daten in und aus Modul über Schnittstellenfunktionen

- Datenabstraktion
  - Anlegen spezifischer, benutzerdefinierter Typen ($\hat{=}$ abstrakte Datentypen)

  $\rightsquigarrow$ viele verschiedene Schnittstellenfunktionen
  - Paradigma: *"Entscheide, welche Typen Du willst; stelle eine komplette Menge von Operationen zur Manipulation eines jeden Typs zur Verfügung."*
  - Problem: statische Typen oft nicht flexibel genug für geringe Erweiterungen
- objektorientiertes Programmieren [OOP]
  - Zusammenfassung von Datentyp (Daten: Attribute) und zugehörigen Operatoren (Methoden) in einer Struktur (Klasse) mit eigenem Typ.
  - Paradigma: *"Entscheide, welche Klassen Du brauchst; stelle für jede Klasse eine komplette Menge von Operationen zur Manipulation zur Verfügung; mache Gemeinsamkeiten mit*

anderen Klassen via Vererbung explizit."

Teile der folgenden Abschnitte sind einem Referat von Sebastian Schmidt vom 17.06.2002 entnommen.

8.1.2 OOP – Allgemein

8.1.2 (a) Grundprinzipien der OOP

- Kapselung: Zugriff auf Attribute nur über Methoden der Klasse, kein direkter Zugriff auf Attribute erlaubt.
- Vererbung: Die abgeleitete Klasse erbt die Attribute und Methoden ihrer Basisklasse(n). Spezialisierung ist dabei möglich durch
  - Überschreiben von Methoden und
  - Hinzufügen neuer Methoden und Attribute.

- Klassen: (benutzerdefinierter) Datentyp; Struktur aus
  - Daten [Members] und
  - Funktionen [Methoden], um diese Daten zu manipulieren
- Member/Elemente/Eigenschaften: typischerweise unter einem Obergesichtspunkt gruppierte Daten
- Methoden: Funktionen, um die Members einer Klasse zu manipulieren
- abgeleitete Klassen:
  - Erweiterung bestehender Klassen, indem die bisherigen Eigenschaften einfach übernommen (*vererbt*) werden und neue hinzugefügt werden.
  - Methoden der ursrpünglichen Klasse können ohne Typumwandlung auf die Member der abgeleiteten Klasse zugreifen

- Instanz / Objekt: Variable vom Typ einer Klasse
- abstrakte Klassen / virtuelle Methoden:
  oft ist es aus Hierarchiegründen sinnvoll eine abstrakte "Urvater"-Klasse zu definieren, deren Methoden aber noch nicht sinnvoll zu definieren sind und erst bei abgeleiteten Klassen sinnvolle Realisierungen besitzen
- Konstruktor: Methode zur Initialisierung eines Objekts vom Typ der entsprechenden Klasse
- Destruktor: Methode zur Löschung eines Objekts vom Typ der entsprechenden Klasse (mit Speicherfreigabe)
- public/private/friends:
  - um die Kapselung zu erleichtern, können bestimmte Methoden und Member, die nur intern gebraucht werden (private), vor dem (fremden) Nutzer der Klasse verborgen bleiben;

  - dieser sieht nur die explizit als ihm zugänglich (public) gekennzeichneten Komponenten;
  - sollen bestimmte Komponenten im allgemeinen wie private behandelt werden, aber für in der Hierarchie nahestehende Klassen zugänglich sein $\rightsquigarrow$ friend–Konzept

- Templates:
  Ebenso wie es sinnvoll sein kann, verschiedene Instanzen ein und derselben Klasse in Form von Variablen zu behandeln, ist es möglich, verschiedene ganz ähnlich zu realisierende Klassen in Form von einem Template zu realisieren; typischerweise können so verschiedene Realisierungen der gleichen Klasse, deren Member aber unterschiedliche Typen haben, z.B. eine Liste mit `double`–, bzw. `char`–Einträgen

8.1.3 OOP – Realisierung in R

8.1.3 (a) Exkurs: "Personen–Kult" zu John M. Chambers

- siehe auch seine Homepage:
  http://cm.bell-labs.com/cm/ms/who/jmc/index.html
- die "farbigen" Bücher
  - "Blue Book":
    Becker et al. (1988), Einführung von `S Version 2`,
  - "White Book":
    Chambers and Hastie (1992), Einführung von `S Version 3`
    — neue Strukturen zur Modellformulierung in `S`
  - "Green Book":
    Chambers (1998), Beschreibung von `S Version 4`
- wichtige Papers:
  "Classes and methods in S." Teil I/II (Chambers (1993a,b)),

auch im Netz: http://cm.bell-labs.com/stat/doc/93.26.ps und http://cm.bell-labs.com/stat/doc/93.27.ps

BEMERKUNG 8.1-1:
Die OOP ist in R historisch bedingt auf zwei verschiedene Weisen realisiert:

- `S3-Classes`:
  basierend auf `S Version 3` — einzige Objektorientierung bis einschließlich `R 1.6.1`
  - noch keine Polymorphie (siehe später)
  - keine formale Klassendefinition möglich
  - Feststellung der Klasse eines Objekts durch das Attribut **class**, s.u.
  - kein klarer Vererbungsmechanismus
  - immer noch wichtig: 80% des (Standard-R-)Codes noch mit S3-Konzept geschrieben

- `S4-Classes`:
  basierend auf `S Version 4` — ab `R 1.7.0` zusätzlich zu `S3-Classes`; größere Umstellungen ab `R 2.0.0`
  - Polymorphie zulässig
  - Grundlage dieser Vorlesung

8.1.3 (b) Abstrakte Datentypen / Klassen

- eine einfache Idee, Programme lesbarer, modularer und leichter modifizierbar zu gestalten
- durch Verwendung eigener Datentypen wird man unabhängig von der konkreten (Rechner–)Darstellung / Implementierung der Daten, man muss nur wissen, welche Operationen zulässig sind
- Beispiel: lm1 ← **lm**(y~x) erzeugt ein Ergebnis vom Typ **lm**.
  - Aufgabe: Ausgabe der Residuen

  - Lösungen:
    a lm1$**residuals**
    b **resid** (lm1)
  - Ansatz (b) ist vorzuziehen, denn er verlangt nicht, dass lm1 eine Liste ist
  - Nach Modifikation / Erweiterung des Typs **lm** ist der erweiterte Typ keine Liste mehr!
  - Ansatz (b) funktioniert aber weiter.

- $\rightsquigarrow$ *"Nicht die Implementation eines Objekts steht im Vordergrund sondern das, was wir mit ihm tun."*
- **CAVEAT:** unterschiedliche Sprechweise: in S heißen Members *slots*
- Instanzierung/Initialisierung eines Objekts durch Konstruktor **new**;
  - Beispiel: x←**new**("circle",**r**=3.3)

– generiert ein Objekt `x` vom Typ / Klasse "`circle`"

8.1.3 (c) Methoden

- in den meisten Umsetzungen des OOP–Konzeps, wie `C++`, `ADA`, `MODULA` sind die Methoden spezielle Member und als solche über die Instanz zugänglich — vgl. z.B. in `C++`: Sei `X` eine Instanz der Klasse `shape`, und diese habe die Methode `rotate`, und die Member `xcoordlist`, `ycoordlist`. Dann rotieren wir die Member von `X` um 20^o durch Aufruf von `X.rotate(20)`; eine solche Herangehensweise heißt in der Terminologie von Bengtsson (2003) *COOP* oder class-object-orientated programming
- in `S` dagegen: *FOOP* (function-object-orientated programming); hier sind die Methoden **nicht** am Objekt, sondern sind *generischen* Funktionen zugeordnet.
- Wie erkennt dann `S`, welche Methode `print` bei einem konkreten Objekt zu verwenden ist?

- Konzepte: *generische Funktion* und *Method Dispatch* [Methoden–Zuordnung]
- über generische Methoden wird Polymorphismus realisiert (mehrere `print`–Methoden für verschiedene Klassen)
- die Liste der Argumente einer Methode (vgl. `args` bei Funktionen) heißt auch `Signatur`
- es sei denn man hat gute Gründe: `...` als Argument in generischer Funktionen, um diese leicht erweiterbar zu halten
- das Method Dispatching erfolgt über die generische Funktion, und zwar wie folgt:
  - bei Aufruf einer generischen Funktion werden alle unter diesem Namen verfügbaren Methoden gelistet
  - gemäß der Klassenhierarchie werden die Methoden angeordnet, und zwar die spezifischste (in der Vererbungs–Hierarchie am weitesten unten stehende) zuerst und dann die weniger spezifischen

  - die spezifischste Methode wird verwendet

- Welche Methode "spezifischer" ist als andere entscheidet S anhand der Klasse der Argumente der generischen Methode — und mit der damit implizierten Klassenhierarchie
- im S3–Klassenkonzept erfolgt das Dispatching nur anhand des **ersten** Arguments, im S4–Klassenkonzept anhand **aller** Argumente
- im S3–Klassenkonzept ist das Dispatching sehr informell; Objekte können ihre Klasse ändern oder auch extern umgewandelt werden.
  Im S4–Klassenkonzept ist die Klasse fix!

8.1.3 (d) Klassen und Vererbung im S3–Klassenkonzept

- Jedes R-Objekt speichert seine Klasseninformation in einem String-Vektor. Das erste Element ist die Klasse des Objekts, das zweite die Vaterklasse, ...
- Setzen des Klassenvektors: **class** `(<obj>) <- <classvector>` (Löschen durch `NULL`.)
- Lesen des Klassenvektors: **class** `(<obj>)`
- **unclass** `(<obj>)` gibt eine Kopie von `(<obj>)` zurück, bei der die Klasseninformation gelöscht ist.
- üblich: Anlegen von *Constructor*en zum Erzeugen von Objekten von S3-Klassen
- Vererbung / Method dispatch geschieht nur durch Namenskonvention $\rightsquigarrow$ unsauberer Programmierung Tür und Tor geöffnet

- Konflikt / Zweideutigkeit: foo.bar.baz wird sowohl als Methode bar.baz der Klasse foo als auch als Methode baz der Klasse foo.bar interpretiert
- Methoden können auch direkt aufgerufen werden (ohne generische Methode) $\rightsquigarrow$ Check der Argument-Typen notwendig

8.1.3 (e) Klassen und Vererbung im S4–Klassenkonzept

- viel umfassender als S3–Klassenkonzept
  - explizite Definition von Klassen und Spezifikation von Vererbung
  - generische Methoden führen Buch über spezifische Methoden
  - multiple Zuordnung [multiple dispatch] möglich

- Beiträge von Bob Gentleman auf der Homepage des "S Programming Workshop" am Department of Statistics an der University of Auckland, 13. und 14. Februar 2003,

  http://www.stat.auckland.ac.nz/S-Workshop/

  - die Slides:

    http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/Methods.pdf

  - eine Kurzzusammenfassung:

    http://www.biostat.harvard.edu/courses/individual/

    /bio271/lectures/L11/S4Objects.pdf

- die R–Manuals

  - R Language Definition:

    http://cran.r-project.org/doc/manuals/R-lang.pdf

  - Writing R Extensions:

    http://cran.r-project.org/doc/manuals/R-exts.pdf

8.1.4 Befehle: Klassen im S4–Klassenkonzept

8.1.4 (a) Klassendefinition: Syntax und Beispiele

- Verwendung der Bibliothek **methods**
- mithilfe der Funktion setClass
- diese hat als Argumente (in dieser Reihenfolge)
  - `Class`: ein Character-String als Name der Klasse
  - `representation`: Namen und Typen der Slots
  - `prototype`: normalerweise eine Liste mit Default–Slotbelegungen; spezifiziert, wie eine neue Instanz dieser Klasse bei der Initialisierung aussehen soll; kann auch über eine `initialize`–Methode für die Klasse kontrolliert werden

- `validate`: eine Funktion, die überprüft, ob es sich bei einer Instanz um eine zulässige Wertebelegung der Klasse handelt
- `where`: in welchem `environment` soll die Klassendefinition abgelegt werden?
- `contains`: welche Klassen sind Ahnen dieser Klasse? Reihenfolge relevant für Dispatch.
- `package`: optional: welchem Paket (`package`) soll die Klasse zugeordnet werden?
- `formal`: Soll eine formale Definition verlangt werden?
- `sealed`: Falls `TRUE`, wird diese Definition gegen Überschreiben, i.e. gegen Neuzuordnung dieses Klassennamens einer alternativen Klassendefinition per `setClass`, geschützt
- `removeSubclassLinks`: Wenn eine Klasse gelöscht wird, werden alle Verknüpfungen anderer Klassen zu dieser Klasse ungültig. Sofern dieses Argument nicht auf `FALSE` gesetzt

wird, sucht removeClass alle solchen Verknüpfungen ab und löscht diese. Man kann dieses Argument auch weglassen, oder auch die Positionen in der Suchliste spezifizieren, die bei dieser Suche nach Verknüpfungen durchlaufen werden sollen — per default werden alle mit `attach` in die Suchliste aufgenommenen Objekt–tables durchlaufen

  - `resetSubclasses`: soll resetClass auch alle bekannten Unter/Nachfahren–Klassen zurücksetzen? normalerweiseTRUE

- Beispiel: eine Struktur für spezifische Messungen der Gestalt $(x, y(x))$
  setClass("track", representation(x="numeric", y="numeric"))
- Beispiel (fortges.):
  eine abgeleitete Klasse, die zusätzlich noch einen Slot `smooth` hat
  setClass("trackCurve", representation("track",smooth="numeric"))
- Beispiel (fortges.): eine Klasse, bei der simultan analog zu

matplot mehrere Kurven verarbeitet werden können; hier ist ein Prototyping sinnvoll:

R–Beispiel 8.1-2 [eine trackMultiCurve-Klasse]:
(aus Chambers (1998)/R-Hilfe)

```
setClass("trackMultiCurve", representation(
            x="numeric",y="matrix",smooth="matrix"),
            prototype=list(x=numeric(),
                           y=matrix(0,0,0),
                           smooth=matrix(0,0,0)))
```

- offenbar ist "trackMultiCurve" kein direkter Nachfahre von "trackCurve"; damit aber dennoch die Methoden von "trackCurve" für "trackMultiCurve" zur Verfügung stehen, wenn die Zahl der Spalten von `y` und `smooth` gleich 1 ist, müssen wir ein **bedingtes Casting** definieren:

R–Beispiel 8.1-3 [bedingtes Casting]:

```
setIs("trackMultiCurve", "trackCurve",
      test = function(obj)#
             {ncol(slot(obj, "y")) == 1},
      coerce = function(obj) {
         new("trackCurve",
             x = slot(obj, "x"),
             y = as.numeric(slot(obj,"y")),
             smooth = as.numeric(slot(obj,
                      "smooth")))
      })
```


- Generierung einer Instanz von "trackCurve":

  R–Beispiel 8.1-4 [Instanzierung]:

```
x0 ← 1:10
y0 ← sin(1:10)+0.3*rnorm(10)
ys ← spline(x0,y0)$y
mycurve ← new("trackCurve",x=x0,
                y=y0, smooth=ys)
```

8.1.4 (b) Zugriff auf Slots im S4–Klassenkonzept

- Typen und Namen der Slots eines Objekts bekommt man mit `getSlots(<class>)` bzw. `slotNames(<class>)`
- Zugriff auf Slots eines Objekts immer mit `@` möglich, z.B. gibt `mycurve@x` den Slot `x` von `mycurve` aus, und `mycurve@y←2*y0` modifiziert den Slot `y` von `mycurve`

- **ABER:** direkter Zugriff auf die Slots ist ein **Verstoß** gegen das Prinzip der Datenabstraktion! — wir hängen nun von der konkreten Implementation der Klasse ab!
- Verdeutlichung anhand eines Beispiels:
  - wir betrachten eine Klasse, die das Konzept "Dreieck" umsetzt; x sei eine Instanz davon
  - ein Dreieck kann man auf verschiedene Arten darstellen:
    * Position aller drei Ecken im $\mathbb{R}^2$
    * Position zweier Ecken im $\mathbb{R}^2$ und deren 2 Winkel
    * Längen der 3 Seiten und Angabe des Punktes a
    * ....
  - es sollen gewisse Berechnungen mit Dreiecken durchgeführt werden
  - Wenn wir auf den Flächeninhalt mit x@area zugreifen, muss die Klasse mit einem solchen Slot implementiert sein.
  - Wenn wir stattdessen area(x) verwenden, können wir die

Methode (mit ein und demselben Namen!) für die verschiedenen Darstellungen von Dreiecken implementieren; im Beispiel einer Implementierung mit Slot `area` könnte das dann so aussehen: setMethod("area","triangle",x@area)

8.1.4 (c) Virtuelle Klassen

- eine *virtuelle Klasse* ist eine Klasse, von der keine Instanzen gebildet werden können
- Zweck: eine gemeinsame Struktur für verschiedene Klassen zur Verfügung stellen, von denen Instanzen gebildet werden können
- Generierung einer virtuellen Klasse:
  - entweder: kein `representation`–Argument in setClass angeben
  - oder: die Klasse VIRTUAL ins `representation`–Argument einschließen

- Konsequenzen
  - die Methoden der virtuellen Klasse stehen allen abgeleiteten Klassen zur Verfügung
  - ein Slot einer neuen Klasse kann vom Typ der virtuellen Klasse sein $\rightsquigarrow$ Polymorphismus, i.e. eine Klasse kann mehrere "Väter" haben
  - die Slots der virtuellen Klasse stehen allen abgeleiteten Klassen zur Verfügung
- Beispiel: die Klasse "vector" ist virtuell und hat als abgeleitete Klassen "getypte" Vektoren, also **character**, **numeric**
  siehe auch getClass("vector"), und getMethods("length") zeigt die für die virtuelle Klasse definierte Methode und wie diese überladen wird

8.1.4 (d) Anwendung virtueller Klassen I: rekursiv definierte Klassen

- es soll ein Dendrogramm realisiert werden, also ein Baum mit zusätzlichen Attributen
- genauer soll es einen Wurzelknoten `wknot`, innere Knoten `iknot` und terminale Knoten `tknot` geben
- der gemeinsame Vorfahre dieser drei Knotentypen sei `knot`

Chambers: Rekursivität keine gute Idee, weil die "whole object" Perspektive nicht möglich ist $\rightsquigarrow$ kein effizienter vektorwertiger Zugriff auf die Knotenelemente eines Baums möglich

- Realisierung:

R–Beispiel 8.1-5 [eine Dendrogramm–Klasse]:

```
setClass("knot")
setClass("wknot",representation(links="knot",
         rechts="knot", Hoehe="numeric"),
         contains="knot")
setClass("iknot",representation("wknot",
         vater="knot"), contains="wknot")
setClass("tknot",representation(label="character",
         value="numeric",Hoehe="numeric",
         vater="knot"), contains="knot")
```

8.1.4 (e) Anwendung virtueller Klassen II: gemeinsame Oberklasse

- oft soll ein Slot je nach Situation entweder leer sein oder eine Liste enthalten

- Problem: NULL ist selbst keine Liste! — daher so ohne weiteres kein gemeinsamer Slot möglich
- Lösung: gemeinsame Oberklasse für NULL und **list**
- dafür ab R 1.8.0 setClassUnion
- Realisierung:

R–Beispiel 8.1-6 [die listOrNULL–Klasse]:

—vgl. Chambers (1998)

```
setClass("listOrNULL")
setIs("list","listOrNULL")
setIs("NULL","listOrNULL");
# ab R 1.8.0 kuerzer:
setClassUnion("listOrNull",c("NULL","list"))
# damit moeglich:
setClass("c1",representation(#
            value="listOrNULL"))
y1<-new("c1",value=NULL);y1
```


```
y2←new("c1",value=list(a=3));y2
```

8.1.4 (f) Initialisierung und Prototyping

- bei der Erzeugung einer Instanz einer Klasse gibt es im Prinzip zwei Mechanismen
  - *Prototyping*: Angabe eines `prototype`–Arguments, i.e. einer Default–Wertebelegung, z.B.

```
setClass("trackMultiCurve", representation(#
            x="numeric", y="matrix",
            smooth="matrix"),
            prototype=list(x=numeric(),
                           y=matrix(0,0,0),
                      smooth=matrix(0,0,0)))
```

  und dann Generierung der Instanz per

```
y1←new("trackMultiCurve")
```


– Bereitstellung einer *Initialisierungsmethode*:

R–Beispiel 8.1-7 [eine Initialisierungsmethode]:

```
setMethod("initialize", "trackMultiCurve",
          function(.Object,x){#
              .Object@x<-x
              .Object@y<-matrix(c(sin(x),cos(x)),
                                length(x),2)
              .Object@smooth<-matrix(c(sin(x),
                 cos(x)),length(x),2)
              .Object}
          )
mycurve<-new("trackMultiCurve",x=0:100)
```


8.1.5 Befehle: Methoden im S3– und S4–Klassenkonzept

Erinnerung:

- Methoden sind in R Funktionen, die man mit unterschiedlichen Argumenttypen aufrufen kann.
- *Generic Functions* sind Funktionen, die in Abhängigkeit von der Klasse ihrer Attribute eine Funktion aufrufen, die die eigentliche Arbeit übernimmt.
- Beispiele: **plot**, **print**, **summary**

8.1.5 (a) Methoden im S3–Klassenkonzept

- Aufruf einer Methode: direkt oder über eine generische Funktion
- **methods(class = c)** gibt alle S3-Methoden der Klasse `c` als Liste zurück.

- Anlegen der generischen Funktion:
  - schreibe eine Funktion `<name>(<obj>)`
  - in dieser Funktion benutze **UseMethod**`(generic=«name>")`
    — Funktionsweise:
    Wird Funktion `<name>` mit Argument `<obj>` aufgerufen und besitzt `<obj>` den Klassenvektor **c**`(class1", class2")`, so versucht **UseMethod** (in dieser Reihenfolge) eine der Funktionen `<name>.class1`, `<name>.class2` oder `<name>.default` aufzurufen. Falls keine dieser Funktionen existiert, wird ein Fehler ausgegeben.

- Anlegen einer spezifischen Methode für Funktion `<name>` und S3-Klasse `<class>`
  - schreibe eine Funktion `<name>.<class>(<obj,....>)` (Namenskonvention verbindlich!)
  - sei `<mclass>` Mutterklasse von `<class>`
  - soll innerhalb `<name>.<class>` auch der Code der Methode `<name>.<mclass>` abgearbeitet werden, so ist dies möglich mit **NextMethod**`(generic=«name>")`
    — Funktionsweise:
    * die Argumente der aufrufenden Funktion `<name>.<class>(<obj,....>)` werden übergeben, auch ihre lokalen Variablen bleiben gültig.
    * **Vorsicht:** Nach Abarbeitung der mit **NextMethod** aufgerufenen Methode kehrt das Programm nicht in die aufrufende Methode zurück, der auf **NextMethod** folgende Teil wird also übersprungen!

8.1.5 (b) Generische Funktionen in S4

- Definition: mit setGeneric
  - Argumente:
    - name: Zeichenkette; Name der generischen Funktion
    - def: eine Funktionsdefinition; erforderlich, falls es bisher noch keine Funktion mit Namen name
    - weitere Argumente: hier nicht; siehe Hilfe zu setGeneric
  - Beispiel

```
setGeneric("myFoo", function(object)
                    standardGeneric("myFoo"))
```

  - dabei gibt standardGeneric("myFoo") einen Fehler aus, wenn keine Methode "myFoo" für den Typ des Arguments definiert ist
- Deklaration spezifischer abgeleiteter Methoden für einzelne Argumenttypen mit setMethod

- Argumente:
  - `f`: Der Name der generischen Funktion als Character-string
  - `signature`: ein Satz formaler Argumentnamen von `f` mit den entsprechenden Klassennamen als Character-strings; kann auch nur ein Vektor von Klassennamen sein; dann entspricht der erste Name dem Typ des ersten Arguments, der zweite dem des zweiten Arguments usw.
  - `definition`: Eine Funktionsdefinition, die verwendet wird, wenn die Argumente im Aufruf von `f` mit denen der Klassen im `signature`–Argument zusammenpassen, — entweder direkt oder per Vererbung
  - weitere Argumente: hier nicht; siehe Hilfe zu setMethod
- R–Beispiel 8.1-8:

```
setMethod("myFoo","character",function(object)
            print(object))
myFoo(a)  # gibt a aus
myFoo(1)  # Fehler da kein Character
```


– R–Beispiel 8.1-9 [ein plot-Befehl]:

```
## Plot--Methoden fuer track Objekte
##
## zuerst mit nur einem Objekt als Argument:
setMethod("plot", signature(x="track",
                            y="missing"),
    function(x,  y, ...)#
        plot(slot(x, "x"), slot(x, "y"), ...)
)
## nun: plot die Daten aus dem track Objekt auf
## der y-Achse gegen was auch immer auf der
## x-Achse
setMethod("plot", signature(y = "track"),
 function(x, y, ...) plot(x, slot(y, "y"), ...)
)
## und entsprechend mit  track--Daten
## auf der x-Achse
```


```
## dabei benutzt: Kurzform der
##          Signaturspezifikation
setMethod("plot", "track",
  function(x, y, ...)#
       plot(slot(x, "y"), y,  ...)
)
```

- Löschen von Methoden und generischen Funktionen mit removeGeneric, removeMethod, removeMethods — letzteres zum Löschen ganzer Gruppen von Methoden

8.1.5 (c) Zugriffsmethoden — Accessor Functions

- um den unmittelbaren Zugriff auf Slots mit @ zu vermeiden ⇝ Zugriffsfunktionen
- Beispiel: Zugriff (noch ohne Modifikationsmöglichkeit) auf Slot y von trackMultiCurve via Methode y

R–Beispiel 8.1-10 [Zugriffsmethode]:

```
if(!isGeneric("y")){
    if(is.function("y"))
        fun←y
    else  fun←function(object)
            standardGeneric("y")
    setGeneric("y",fun)}
setMethod("y","trackMultiCurve",
            function(object)object@y)
mycurve←new("trackMultiCurve",x=0:100)
y(mycurve)
```


- Beispiel: Überladen des +–Operators für Zeichenketten

```
setMethod("+",c("character","character"),
          function(e1,e2)#
              paste(e1,e2,sep=""))
```

- R–Beispiel 8.1-11 [Indexoperator]:
  — siehe auch Übung/Hilfe:
  Obwohl "[" und "length" keine gewöhnlichen, sondern . **Primitive** Funktionen sind (d.h. Code nicht sichtbar!), kann man sie trotzdem überladen:

```
setMethod("[", "track",
   function(x, i, j, ..., drop) {
     x@x <- x@x[i]; x@y <- x@y[i]
     x})
```


```
t1 ← new("track", x=1:20, y=(1:20)^2)
plot(t1[1:15])
```

8.1.5 (e) Ersetzungsmethoden

- Problem: in S eigentlich kein "Pass by Reference" zur Übergabe von Argumenten, sondern "Pass by Value", i.e. es werden lokale Kopien angelegt

$\rightsquigarrow$ eigentlich keine Modifikation von Argumenten möglich

- aber: in S geschieht alles über Funktionen! Wie ist also dann die Zuweisung x[1]←10 organisiert?
  - x[1]←10 wird intern umgesetzt in
    x←**do.call**("[←", **list**(x,1, value=10))
  - dabei wird zuerst x kopiert und der Wert des ersten Elementes wird auf 10 gesetzt

  - die Funktion `[←` gibt dann ein Objekt vom gleichen Typ wie `x` zurück, nur eben mit den gewünschten Änderungen
  - zuletzt verknüpft (rebinds) der Auswertungsmechanismus in S das Symbol `x` mit diesem neuen Wert

- auf diese Weise "Pass by Reference" nachgeahmt
- für Ersetzungsmethoden ist die Strategie ähnlich:
  - zuerst muss eine generische Funktion erzeugt werden — mit einem `←`–Suffix
  - dann muss man die Ersetzungsmethode definieren
  - dabei heißt das letzte Argument `value`, und die Methode liefert eine modifizierte Kopie des ersten Arguments zurück
  - beide Argumente müssen dabei angegeben werden, damit die Methode funktioniert.

- R–Beispiel 8.1-12 [Ersetzungsmethode]:

```
setGeneric("y←",function(x,value) #
                standardGeneric("y←"))
setReplaceMethod("y","trackMultiCurve",
                 function(x,value)
                          {x@y←value
                           x}
                 )
y(mycurve)←y(mycurve)−2
```

- anderer Trick, "Pass by Reference" nachzuahmen:

  eval.parent(substitute(x@<slot>←value))

  Dabei gibt **substitute** zu einen nicht ausgewerteten Ausdruck im Argument den *Parsing-Tree* zurück; dieser wird dann mit **eval.parent** im aufrufenden `frame` ausgewertet; vgl. auch Beispiel 6.6-1

- R–Beispiel 8.1-13 [Simulationsbeispiel]:

```
setMethod("simuliere",
          signature(x="simulation"),#
          function(x)#

           {set.seed(x@seed)
            if(x@runzahl*x@samplesize>0)
              eval.parent(substitute(#
                x@Daten<-matrix(rnorm(#
                  x@runzahl*x@samplesize),#
                  x@runzahl,x@samplesize)))
            return(invisible())
           }
         )
```


8.1.5 (f) elementweise Funktionen, arithmetische und andere Operatoren

- in S möglich: Gruppierung von Methoden, um diese als Gruppe bestimmten Klassen zur Verfügung zu stellen (c.f. 8.1.5 (g))
- wichtige Gruppe: die Gruppe der `Math`–Funktionen
  - genauer: `Math` ist die Gruppe der elementweise auf Datenstrukturen ausführbaren Operationen, wie z.B. `"+"` (das unäre!), **sin**, **is.null**, ...
  - R–Beispiel 8.1-14 Math–Funktionen:
    - Ziel: `Math`–Funktionen für Simulationsklasse aus Übung
    - Umsetzung: `Math`–Funktionen wirken auf Slot `Daten`

```
setMethod("Math","simulation",
function(x){#
        x@Daten=callGeneric(x@Daten)
        x})
```


- weitere Gruppen:
  - `Math2`
    - zusätzlich zu `Math`: ein zweites Argument `digits`, das die Genauigkeit steuert;
    - Beispiele: **round**, **signif**
  - `Summary`
    - aggregieren (meistens) numerische Werte
    - Beispiele: **sum**, **summary**, **max**
    - **summary** fasst auch nicht numerische Werte zusammen
    - Problem: Anwendung einer Funktion auf eine Liste typ-heterogener Objekte?
    - Lösung: `S` wendet die Funktion dann rekursiv elementweise mit der entsprechenden Methode an
- arithmetische Operatoren
  - `S` benutzt *"infix"*–Notation, d.h. `x*y` anstelle der funktionalen *"postfix"*–Notation `"*"(x,y)` wie z.B. `Lisp`

- aber nur eine Frage des Interpreters; realisiert sind diese Funktionen genauso wie gewöhnliche
- jeder solche Operator ruft eine Funktion mit Namen des "infix"–Operators in Anführungszeichen auf, also z.B. wird `x*y` intern umgesetzt in `"*"(x,y)`
- Gruppen solcher zweistelligen Operatoren:
  * `Arith` — Beispiele: `+` (binär!), `/`, `^`
  * `Compare` — Beispiele: `>`, `>=`, `!=`, `compare`
  * `Logic` — Beispiele: `&`, `!`, `|`
  * `Ops` = `Arith`,`Compare`,`Logic`
- Problem: Was tun, wenn der erste Operand von anderem Typ wie der zweite?
- genauso: falls der zweite Operand fehlt ("**missing**"), dann sollte der Operator als unär interpretiert werden können, wenn dies sinnvoll ist

⇝ finden einer gemeinsamen Mutterklasse, Casting, siehe Abschnitt 8.1.8

Chambers: Operatoren sollten sinnvoll sein, wenn einer der beiden Operanden numerisch $\Longrightarrow$ `"+"` für Strings — wie in `JAVA` — keine gute Idee

8.1.5 (g) eigene Gruppen generischer Funktionen

- Ziel: Anlage eigener Gruppen analog zu `Math`–Funktionen
- wichtig: alle Funktionen in der Gruppe müssen die gleichen formalen Argumente (und Namen!) besitzen — es findet kein weiteres matching zwischen dem Aufruf und der Ausführung der Methode statt!
- Anlegen einer Gruppe durch:
  `setGroupGeneric("<Grpname>", function(<formaleArgs>)NULL)`
- Eintragen als Mitglied einer Gruppe durch:
  - `setGeneric("<Fktname>",group="<Grpname>")`
  - falls die Funktion schon als generisch deklariert ist:
    `setGroup("<Fktname>","<Grpname>")`

- Anzeige aller Gruppenmitglieder mit
  `getGroupMembers("<Grpname>")`

8.1.6 Befehle: Anfragen, welche Methoden wie existieren

- mit exaktem match:
  - Gebe Methode zu vorgegebener Signatur in entsprechender Datenbank zurück: $\rightsquigarrow$
    `getMethod("<Fktname>", signature(<Signatur>),`
    `[where=<Database>|frame=<frame>])`
  - Gibt es Methode zu vorgegebener Signatur in entsprechender Datenbank? $\rightsquigarrow$
    `existsMethod("<Fktname>", signature(<Signatur>),`
    `[where=<Database>|frame=<frame>])`
  - Finde Methode zu vorgegebener Signatur: $\rightsquigarrow$
    `findMethod("<Fktname>", signature(<Signatur>))`

- Gebe alle Methoden zu einer generischen Funktion in entsprechender Datenbank zurück: ⇝
  `getMethods("<Fktname>",[where=<Database>|`
  `frame``=<frame>])`
- Zeige alle Methoden zu einer bestimmten generischen Funktion ⇝
  `showMethods("<Fktname>",[where=<Database>|`
  `classes=<clslist>|, includeDefs=<T|F>|,`
  `inherited=<T|F>|, printto=<connection>])`
  Dabei ist `<clslist>` eine Liste mit Klassen, auf die die Suche beschränkt werden kann, mit `includeDefs=T` lässt man sich die Bodies der Methoden mit ausgeben, mit `inherited=T`, werden die neu als ererbt erkannten Methoden mit angegeben, mit `<connection>` können wir eine andere Ausgabe als die Konsole wählen —meist ein File.

- mit Ableitung / Dispatch (i.e. `extends`– und `group`–Relationen)
  - Gebe Methode zu vorgegebener Signatur in entsprechender

Datenbank zurück: ⇝

```
selectMethod("<Fktname>", signature(<Signatur>),
             [where=<Database>|frame=<frame>])
```

– Gibt es Methode zu vorgegebener Signatur in entsprechender Datenbank? ⇝

```
hasMethod("<Fktname>", signature(<Signatur>),
          [where=<Database>|frame=<frame>])
```

- In welcher Datenbank ist eine Methode als generisch deklariert? ⇝ `findGeneric("<Fktname>")`
- Ist Funktion als generisch deklariert? ⇝

```
isGeneric("<Fktname>")
```


8.1.7 Befehle: Versionsmanagement

[noch nicht in R realisiert; nur in Chambers (1998) konzipiert]

- praktisch nicht vermeidendes Problem:
  möchte Instanz einer Klasse einlesen, habe aber zwischenzeitlich die Klassendefinition geändert
- Default Lösung: S liest die Instanz in einer "unklassierten" Form, d.h. in einem Typ, der die gleichen Daten enthält wie das ursprüngliche Objekt, aber nicht mehr von diesem Typ ist
- besser: S bietet Tools, um verschiedene Versionen von Klassendefinitionen zu verwalten; diese umfassen
  - Verwaltung eines Objekts vom Typ version
  - diese enthält *Meta–Information* wie
    - sukzessive " representation "s einer Klasse

genauer: **unclass**(<Instanzname>)

    * Methoden zur Konvertierung von einer Klassendefinition in eine andere — oft automatisch beim update erstellbar, bzw. falls nicht vom Entwickler bereitzustellen

– Versionsmanagement ist optional aber sehr empfohlen
– Anmeldung einer Klasse zum Versionsmanagement mit `setClassVersion ("<Classname>")`
– Anmeldung eines Klassenupdates zum Versionsmanagement mit `setClassVersion ("<Classname>")`
– Umwandlung der Objekte vom alten Typ in den neuen Typ mit `updateObjects("<Classname>")`
– bei einer einfachen Erweiterung wird die `updateObjects`Methode gemäß folgender Heuristik automatisch erstellt
  1. Slots, die in einer alten Version vorhanden waren und in der neuen nicht mehr, werden außer acht gelassen
  2. neuhinzugekommene Slots werden mit den default–Werten

aus dem `prototype`–Argument initialisiert

3. wenn ein Slot, der in beiden Versionen vorhanden ist, seine Klasse geändert hat, wird versucht, diesen mit einem Aufruf der **as**–Methode zu casten

– bei jedem Aufruf von setClassVersion werden außerdem Datum und ein default–Kommentar zu den Änderungen abgelegt; diesen kann man im Aufruf auch selbst setzen

– bei substanzielleren Änderungen muss der Entwickler die update–Methode im setClassVersion –Argument `method` bereitstellen, in der er ein Objekt, das aus **unclass**`(<Instanzname>)` hervorgeht, in den neuen Typ konvertiert

– weiterhin hat setClassVersion ein Argument `which` das steuert, in welche Version automatisch gecastet werden soll — unbedingt angeben, falls man nur eine Testversion anmelden möchte!

8.1.8 Befehle: Typüberprüfung zur Laufzeit und Casting

- mit den Vererbungsmechanismen und den ausgeklügelten Castings ist es faktisch nicht möglich, zum Zeitpunkt der Programmierung eine Typ–Prüfung durchzuführen

$\rightsquigarrow$ Typüberprüfung zur Laufzeit

8.1.8 (a) Validitäts–Checks

- bei (permanenten) Zuweisungen wird automatisch geprüft, ob ein Objekt der gültigen Klassendefiniton genügt $\hat{=}$ automatischer Validitäts-Check
- auch: expliziter Validitäts-Check durch validObject
- per default: nur Prüfung der Typen der "representation" der Klasse

- oft genauere Prüfung sinnvoll $\rightsquigarrow$
  - Angabe einer validity –Methode als Argument bei der Klassendefinition
  - oder spätere Angabe mit einem Aufruf von setValidity
- die validity –Methode liefert entweder TRUE oder eine String–Message mit einer Problembeschreibung
- Beispiel 8.1-15 [Validität für die Matrixklasse]:
  - in der Arrayklasse wird geprüft, ob die Elemente des Dimensions-Slots .Dim und positiv sind, und mit der Länge des **data**–Arguments zusammenpassen...
  - Angabe der validity –Methode durch
    setValidity ("array", validArray)
  - dabei verwenden wir für validArray folgende Definition:

```
validarray<-function(object){#
d<-object@.Dim
if(!all(d>=0))
    return("Negative Elemente in dim!")
else if (prod(d)!=length(object@.Data))
  return("Falsche Datenlaenge")
dn<-object@.Dimnames
if(length(dn)>0&&length(dn)!=length(d))
  return("Falsche Zahl an Dim-Namen!")
else {
    dn1<-sapply(dn,length)
    if(any(dn1>0 & dn1!=d))
        return("Falsche Elementlaengen!")
        }
return(TRUE)
}
```


  - Achtung: eigentlich Verwendung von `@` schlechter Stil, aber hier genau angebracht!
    Denn: hier soll explizit die Implementation überprüft werden!

- S wird diese neue validity –Methode nun bei jeder (permanenten) Zuweisung verwenden
- um bei einem fehlgeschlagenem Validitäts-Check bei einer Zuweisung keine Daten zu verlieren, wird das Objekt zwar zugewiesen aber mit **unclass** "unklassiert"
- generelle Frage: *Wann soll geprüft werden?*
- In S: bei (permanenten) Zuweisungen, da sonst zu zeitaufwendig

$\rightsquigarrow$ bei Übergabe eines Objekts als Argument einer Funktion **keine** erneute Prüfung!!

8.1.8 (b) Spezifizieren von `is`–Relationen — der `setIs` Befehl

- Situation:
  - die Signatur einer Methode enthält ein Argument `arg` der Klasse `c1`;
  - wir wollen die Methode auf eine Instanz `X` der Klasse `c2` anwenden

  $\rightsquigarrow$ `X` muss — möglichst automatisch — in Klasse `c1` gewandelt / gecastet werden
- Sprechweise: in S heißt dies *coercing*
- beim automatischen Casting muss erst dessen Zulässigkeit überprüft werden
- statische Variante: **is**
  — gibt TRUE zurück, falls `c1` eine Oberklasse von `c2` ist

- oft unflexibel:
  - die Oberklasse "weiß" gar nicht, dass sie eine Oberklasse ist
  - ein Casting ist nur in manchen wichtigen speziellen Wertbelegungen des Objekts möglich — vgl. Beispiel 8.1-3
- erster Punkt: explizite Deklaration einer `is`–Relation durch `setIs`
- zweiter Punkt $\rightsquigarrow$ S erlaubt *bedingtes* Vererben
- dazu explizite Angabe eines "bedingten" `is` durch `setIs` unter Angabe eines Arguments `test` — eine Funktion die testet, ob die Bedingung zum Casten erfüllt sind, dann `TRUE` zurückgibt und andernfalls einen Fehler ausgibt
- **WICHTIG**: die Testfunktion muss einen einzigen logischen Wert zurückgeben — niemals Vektoren oder `NA`'s
  - $\rightsquigarrow$ keine Verwendung von `==`
  - $\rightsquigarrow$ Verwendung von **any**, **all**, identical
- die Relation "Klasse `c1` ist Oberklasse zu Klasse `c2`" wird in S

durch die Funktion extends beschrieben

$\rightsquigarrow$ um alle Oberklassen einer Klasse im aktuellen Suchpfad zu bekommen: `extends("<Classname>")`

- extends ist transitiv!
- keine einfache automatische Typumwandlung möglich $\rightsquigarrow$ eigene Casting–Methode als Argument coerce der Funktion setIs

8.1.8 (c) explizites Casting: as–Relationen

- generelle Methode zur Typumwandlung
  `as(<objname>,"<Classname>")`
- um eine eigene **as**–Methode zum expliziten Casten zur Verfügung zu stellen: Deklaration als as–Relationen mit setAs

8.1.9 Erfahrungen mit S4-Klassen

dieser Abschnitt wurde von *Matthias Kohl* beigetragen

- langsames Anlegen von S4-Objekten
  - Anlegen neuer S4-Objekte ist ziemlich aufwendig (evtl. inkl. einem validity check)

TIP: innerhalb von Funktionen mit S4-Objekt einer bestimmten Klasse als Argument, die neues Objekt eben dieser Klasse erzeugen:

- besser: kein neues Objekt erzeugen

  - ∗ sondern: direkt slots des übergebenen Objekts modifizieren —evtl. mit @-Operator und dieses zurückgeben

  ⇝ deutlich schneller

  ⟹ besser bestehende S4 Objekte modifizieren als neue generieren

- schnelleres Dispatchen ("beim zweiten Mal"...)
  - bei Aufruf konkreter Methode wird Methode mit konkreter Signatur im Suchpfad abgelegt (mit Angabe inherited from ...)

BSP "+" im Zusammenhang mit "distr"

⇝ Aufruf dieser Methode ist beim nächsten Mal schneller

- Beispiel-Code

```
require("distr")
showMethods("+")
Norm() + Pois()
showMethods("+")
```

- Probleme:
  - z.B. bei getInfGamma in Paket ROptEst
  - Situation: konkrete Methode mit verschiedenen Signaturen
  - gibt zwei zulässige (unterschiedlich spezfische) Methoden
  - eine passt genau, die andere nur per Vererbung
  - bei erstem Aufruf: genau passende wird verwendet
  - dazwischen: Aufruf der Methode mit anderer Signatur

  ⇝ genau passende wird überschrieben
  - bei nächstem Aufruf mit ursprünglicher Signatur: schlechter passende wird verwendet

- zu showMethods: Wie kommt man an body einer Funktion?
  - S3 Methoden: am Beispiel **print**

```
methods(print)
    # Methoden mit * sind "non-visible"
# oder
methods(class = "anova")
```


```
    # Methoden für bestimmte Klasse
# als Beispiel
print.anova
# oder
getS3method("print", class = "anova")
# für non-visible (in welchem Namespace?)
getAnywhere("print.aov")
    # aha, Namespace "stats"
# also auch
stats:::print.aov
# oder
getFromNamespace("print.aov", "stats")
```

– S4 Methoden: am Beispiel **show**

```
# mit getMethods werden alle Methoden inklusive
# ihrer Definition angezeigt => unübersichtlich
# bei vielen Methoden
```


```
getMethods("show")
# besser
showMethods("show")
# um einzelne Methoden inkl. Definition
# anzuzeigen, kann man folgendermaßen vorgehen
getMethod("show", "traceable")
# oder etwas umständlicher
showMethods("show", classes="traceable",
                    includeDefs = TRUE)
```

- Wann wird/soll validate ausgeführt werden?
  - wegen Geschwindigkeit: in setValidity Methode nur einfache Checks einbauen
  - genauerer Check möglich z.B. über neue generische Funktion mit entspr. Methoden —vgl. checkIC in ROptEst
  - Anlegen der setValidity Methode am besten gleich bei Einführung neuer Klasse

- sonst möglich: Probleme bei tief gehenden Vererbungen

- setIs , setAs —Erfahrung und unerwartetes Verhalten...
  - setIs , setAs noch nicht völlig ausgereift [Stand: `R 2.0.1`]:
    * für nachträgliche setIs –Relation: auch "coerce" Methode (per setAs) überladen!

Achtung bei Verwendung von Namespaces:

- Anlegen setAs-Relation $\Longrightarrow$ benötige exportMethods("coerce")
- Anlegen einer setIs -Relation erfordert zusätzlich exportMethods("coerce←")

* weiteres Problem
  - Ziel: " UnivariateDistrList " der Länge 1 in entspr. Verteilung wandeln

$\rightsquigarrow$ setAs(" UnivariateDistrList ", " UnivariateDistribution ")

- Annahme: Objekt der Klasse " UnivariateDistrList " hat Länge 1 und enthält Objekt der Klasse "Norm".
- Dann: **as** (...) liefert nicht Objekt der Klasse "Norm",

sondern " UnivariateDistribution "

$\Longrightarrow$ setAs für alle Verteilungsklassen einzeln definieren

- initialize oder nicht
  - bei Anlegen einer initialize Methode wird die default-Methode überschrieben, in der jeder einzelne Slot gesetzt werden kann
  - vermeidbar durch Einsatz von *generating functions*.
  - Chambers (1998) empfiehlt deren Einsatz, um Benutzer Generieren von Klassen zu erleichtern
  - in *generating function* / initialize Methode möglich: einfache Validity Checks
  - durch Verwendung der *generating function* bleibt default initialize Methode erhalten (deren Verwendung bei Programmierung manchmal von Vorteil)
  - eigene initialize Methode sinnvoll:
    - will verhindern dass bestimmte Slots explizit beim

Initialisieren gesetzt werden können

* BSP: Verteilungen, bei denen man nur Setzen der Parameter erlauben will, während **r**,**d**,**p** und **q**-Slot garantiert durch base-Methoden belegt sein sollen

8.2 Schreiben eigener Pakete

nach einem Tutorial von T. Lumley und D. Bates in Wien 2003

8.2.1 Wie benützt man R effizient?

8.2.1 (a) Wie speichert man seine Routinen?

- als **Workspace**:
  - beim Starten von R liest dieses das File `.RData`
  - beim Beenden Frage: gesamter Arbeitsspeicher (ohne eingeladene Pakete) als Workspace auf dieses File schreiben?
  - dies ist auch jederzeit mit **save.image()** möglich; wieder einladen mit **sys.load.image()**
- als **Binärdateien**:
  - mit dem **save**–Befehl kann man einzelne Funktionen und Daten in Binärdateien schreiben

  - diese kann man mit **attach** oder **load** wieder einlesen

- als **Quelldateien**

8.2.1 (b) mehrere Projekte

"Extremstrategien":

- jedes Projekt in ein eigenes Verzeichnis
  - mit einem `.RData`–File in diesem Verzeichnis alles Wichtige ständig im Speicher halten
  - wenn Daten und Funktionen in mehreren Projekten gebraucht werden, diese hin- und herkopieren
  - das `.RData`–File ist zentral; alle Quelldateien dienen nur der Dokumentation
- alles als Sourcen speichern
  - für jeden Analyseschritt hat man eine Quelldatei, mit der die Daten eingelesen, verarbeitet und möglicherweise wieder modifiziert abgespeichert werden

– die Quelldatei ist zentral — alle anderen abgespeicherten Dateien dienen nur der Aufwandsminimierung beim Verarbeiten

Empfehlung in R:
Weil Objekte nicht automatisch gespeichert werden (wie in `S-Plus`) besser alles als Sourcen speichern!

8.2.1 (c) Nutzung von R–Output

- unter Windows kann man den Output der Konsole als Textfile abspeichern
- **write.table** produziert formattierten Text und kann im wesentlichen von LaTeX weiterverwendet werden
- im Prinzip ist es möglich, über die `DCOM` Schnittstelle automatisierte Reports in MS WORD zu generieren
- mit dem `XML` package kann man die Ausgabe in `XML` erreichen

8.2.2 das R–packaging System

8.2.2 (a) Warum packages ?

R packages — im folgenden Pakete — erlauben es, Funktionen und Daten zusammen mit ihrer Dokumentation zusammenzufassen.

Bemerkung: Nicht verwechseln!! **library** und **package**; dazu Zhu Wang and Douglas Bates in `R-help, (May 2004)` (aus Paket `fortunes`)

> Z.W.: "I am trying to create a library which uses some Fortran source files [...]"
> D.B.: "Someone named Martin Maechler will shortly be sending you email regarding the distinction between 'library' and 'package' :-)"

Im Unterschied zu anderen Programmiersprachen gilt in S die Konvention: Ein Paket (**package**) wird in einer **library** abgelegt, d.h. **library** ist eine Suchliste –vgl. Abschnitt 1.10.3.

- dynamisches Ein– und Ausladen:
  das Paket beansprucht nur dann Speicher, wenn es benutzt wird
- einfaches Installieren und Aufdatieren:
  die Funktionen, Daten und Dokumentationen werden alle konsistent und an die korrekte Stelle mit einem einzigen Befehl installiert und können entweder innerhalb oder außerhalb R ausgeführt werden

- Anpassung durch Nutzer oder Administrator:
  neben `LAN`–weiten Bibliotheken können einzelne Nutzer individuelle, private Bibliotheken von Paketen haben
- automatische Validierung:
  R stellt Befehle zur Verfügung, um offensichtliche Fehler aufzudecken, und um zu überprüfen, ob eine Dokumentation existiert und ob die angegebenen Beispiele so lauffähig sind
- die meisten Nutzer kennen Pakete von der Grunddistribution von R und von `CRAN`; mit dem R–packaging System können noch viel mehr Leute zu der Entwicklung von R beitragen, wobei gleichzeitig gewisse Mindeststandards garantiert werden
- `Data packages` sind sehr sinnvoll in der Lehre: Datensätze können zusammen mit ihrer Dokumentation und Beispielen verfügbar gemacht werden — c.f. `Devore5`–Paket von D. Bates
- Private Pakete sind extrem nützlich, um oft genutzte Funktionen und Daten zu organisieren und zu speichern

8.2.3 Struktur von R–Paketen

- die Grundstruktur eines R–Pakets ist ein Verzeichnis, das gewöhnlich enthält
  - ein `DESCRIPTION`–File mit einer Beschreibung des Pakets, Autor und Lizenzvereinbarung in einer strukturierte Textdatei
  - ein `INDEX` file, das alle Funktionen und Daten (und optional weitere Informationen) enthält — kann automatisch erzeugt werden!
  - ein `NAMESPACE` file, siehe Abschnitt 8.2.7
  - ein `CITATION` file, in dem man einen BibTEX Eintrag für sein Paket vornehmen kann
  - ein `configure` file und ein `cleanup` file, in dem man unter `Unix` (Bourne-)shell Scripten zum Aufruf vor und (sofern mit Option `-clean` gearbeitet wird) nach der Installation ablaufen lassen will

  - ein `COPYING` file, in dem —falls abweichend von der `GPL`-Lizenz— Regelungen zum Kopieren des Pakets angeben kann
  - ein `man`/–Unterverzeichnis mit Dokumentationsfiles (im `.Rd`-Format, siehe Abschnitt 8.2.6
  - ein `R`/–Unterverzeichnis mit den R–Quelltexten
  - ein `data`/–Unterverzeichnis mit den Datensätzen
  - ein `src`/–Unterverzeichnis mit den Quellfiles für `C`, `FORTRAN` oder `C++` Routinen
  - ein `demo`/–Unterverzeichnis mit ausführbaren Demos zum Paket
- nicht immer enthält es
  - ein `tests`–Unterverzeichnis mit Validierungstests
  - ein `exec`/–Unterverzeichnis mit anderen ausführbaren Programmen (z.B. `JAVA` oder `Perl`–Routinen)
  - ein `inst`/–Unterverzeichnis mit verschiedenen anderen

Dingen wie .ps–Files

  - ein configure–Skript um die Verfügbarkeit anderer benötigter Software zu checken oder um Unterschieden zwischen Betriebssystemen zu handlen

- optional enthält es (ohne dass R es verwendet)
  - README file
  - NEWS file
  - Changelog file
- außer dem DESCRIPTION– und dem INDEX–File sind die meisten Punkte optional
- aber jedes sinnvolle Paket wird eine Dokumentation und mindestens entweder ein R/– oder ein data/–Verzeichnis haben wird.
- **Referenz** zu diesem Abschnitt: Writing R Extensions (2006b, Abschnitt 1)

8.2.4 Aufbau des DESCRIPTION-file

- Elemente:
  es müssen/können Angaben zu folgenden Punkten gemacht werden — in der Form `<Punkt>: <Angabe> <!neue Zeile>`
  - obligatorisch: Package, Title, Version, Author, Maintainer, Description
  - optional: Abhängigkeiten: Depends, Imports, Suggests
  - optional: schnelleres Laden; LazyLoad, SaveImage,
  - optional: SystemRequirements, Date, URL

  Folgende Elemente werden automatisch generiert:
  - Built
  - Packaged
- dabei bedeuten
  - *Date:* Erstellungsdatum, in "JJJJ-MM-TT"-Format

- *LazyLoad:* ein Mechanismus, der ab R Version 2.0.0 zur Verfügung steht: er lässt zu, dass Objekte erst in den Speicher geladen werden, wenn sie gebraucht werden; genaueres siehe Ripley (2004)
- *SaveImage:* ein Mechanismus, der regelt, ob die R Objekte des Pakets aus einem gespeicherten `image` geladen werden —d.h. ein mit **save** / **save.image** abgelegter Workspace, siehe Abschnitt 1.9.1
- *Description:* Eine kurze, umfassende Beschreibung dessen, was im Paket steht (ein Absatz)
- *Maintainer:* ein *einzelner* Verantwortlicher für das Paket
- *License:* unter welcher Lizenz darf das Paket verwendet werden. . .—typischerweise `GPL`
- *URL:* ein Link auf eine Homepage zu diesem Paket
- *Depends:* eine Komma getrennte Liste von Paketen, die das eigene Paket braucht; optional können die Paketnamen von

einem Ausdruck (<= <N>) bzw. (>= <N>) mit der entsprechend benötigten Version gefolgt werden, z.B. `distr(>= 1.3)`

- *Imports:* listet die Pakete auf, deren Namespaces importiert werden sollen, siehe auch Abschnitt 8.2.7
- *Suggests:* gleiche Syntax wie *Depends*; listet Pakete die wünschenswert sind, aber nicht notwendig sind
- *SystemRequirements:* darüber hinausgehende Anforderungen an das System

- Beispiel

```
Package: distr
Date: 2004-03-15
Title: Object orientated implementation of distributions
Version: 1.5
Depends: R(>= 2.0.0), methods, graphics, setRNG
Imports: stats
LazyLoad: yes
SaveImage: no
Author: Florian Camphausen, Matthias Kohl, Peter Ruckdeschel, Thomas
        Stabla
Description: Object orientated implementation of distributions and
        some additional functionality
Maintainer: Peter Ruckdeschel <Peter.Ruckdeschel@uni-bayreuth.de>
License: GPL (version 2 or later)
URL: http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/
Packaged: Thu Apr 15 16:09:02 2004; tom
Built: R 2.0.1; i686-pc-linux-gnu; 2005-03-15 14:14:20; unix
```


8.2.5 Format für Datensätze

Datensätze werden mit dem Befehl **data()** eingeladen; diese können abgelegt sein als

- Textfile (`.txt`, `.tab`, `.csv`)— entweder Komma– oder `whitespace` getrennt
- `S` Quelltext, (`.r`, `.R`) erzeugt durch den **dump**–Befehl
- R–Binärdateien, (`.rda`, `.RData`) erzeugt durch den **save**–Befehl

Der Filetyp wird anhand der Endung bestimmt.

8.2.6 Dokumentation

Das R–Dokumentationsformat lehnt sich an LaTeX an. Details siehe Writing R Extensions (2006b, Abschnitt 2).

R–Beispiel 8.2-1 [eine Beispieldokumentation]:

```
\name{birthday} % Anzeigename des Hilfe-Files
\alias{qbirthday} % die Funktionen,
\alias{pbirthday} %     die hier dokumentiert werden
%einzeiliger Titel der Dokumentationsseite
\title{Wahrscheinlichkeit einer Koinzidenz}
%kurze Beschreibung
\description{
  berechnet approx. Lsg. zu verallgem. Geburtstagspb;
  \code{pbirthday} berechnet die W-keit(Koinzidenz) und
  \code{qbirthday} ....}
\usage{ % wie man die Funktionen aufruft
pbirthday(n,classes=365, coincident=2)
qbirthday(prob=0.5,classes=365, coincident=2)}
```


```
\keyword{distribution}
\concept{elemtary statistical problems}
```


8.2.6 (a) Abschnitte des Hilfefiles

- \name: Anzeigename des Hilfe-Files; nicht notwendig Übereinstimmung mit Filenamen des `.Rd`-Files
- \alias
  - zur Gruppierung der Dokumentation verschiedener ähnlicher, inhaltlich zusammengehörender Objekte; z.B. **rnorm**, **pnorm**, **qnorm**, **dnorm**
  - Kriterium für **R** CMD **check**: Ist jedes Objekt im Paket dokumentiert?
- \arguments: eine Aufzählung der Argumente und ihrer Bedeutung
- \value: Beschreibung des Rückgabewertes
- \details : eine längere Beschreibung der Funktion, falls notwendig

- \references: Literaturhinweise oder andere bibliographische Verweise (v.a. InterNet)
- \seealso: Referenzen zu verwandten Befehlen
- \examples: Anwendungsbeispiele für die Funktion
- \keyword: zur Erstellung eines Index (vorgegebene Schlagwörter —siehe File `KEYWORDS.db` im Verzeichnis `doc` unter dem R-Stammverzeichnis
- \concept: selbstvergebene Schlagwörter
- \section{<**title**>}: Abschnitt mit selbst gewähltem `<title>`

8.2.6 (b) Dokumentation von Datensätzen

- obligatorisch: \docType{**data**}, \usage{**data**(<pkgname>)}, \keyword{datasets}
- **format**: in welchem Format die Daten vorliegen
- **source**: woher die Daten stammen

8.2.6 (c) Formatbefehle zur Textmarkierung à la LaTeX

- Einfügen von Zeilenumbrüchen mit `\cr`, neue Absätze mit einer Leerzeile
- `\emph{<txt>}`, `\bold{<txt>}`, `\strong{<txt>}` zum Hervorheben
- `\squote{<txt>}`, `\dquote{<txt>}`: Zitate mit ‘...’, bzw “...”
- `\code{<txt>}`: R-code; die Zeichen %, {, } müssen durch \ maskiert werden, d.h. man fügt ein ‘\’ vor diese Zeichen
- `\preformatted{<txt>}`: vorformatierter Quellcode/Text
- `\samp{<txt>}`: Beispiel einer Zeichenkette
- `\kbd{<kbd-chars>}`: Eingabe an Konsole
- `\pkg{<pkgname>}`: ein Paket
- `\file{<filename>}`: ein File
- `\email{<filename>}`: eine Email-Adresse

- `\url{<filename>}`: eine URL
- `\var{<varname>}`: metasyntaktische Variable (??)
- `\env{<envname>}`: Umgebungsvariable
- `\option{<optname>}`: Kommandozeilenoption
- `\command{<cmdname>}`: Befehl
- `\dfn{<cmdname>}`: Definition (bei der Einführung eines Begriffs)
- `\acronym{<cmdname>}`: ein Acronym wie `GNU`
- `\cite{<reference>}`: unverlinkte Referenz, z.B. ein Buch
- `\link[<pkg>|<pkg>:<topic>]{<obj>}`: verlinkte Referenz auf die Hilfeseite zu Objekt `<obj>` (zu Paket `<pkg>`, bzw. Thema `<topic>` in Paket `<pkg>`)

8.2.6 (d) Listen-/Tabellenbefehle à la LaTeX

- Listen:
  - Befehle: `\itemize`, `\enumerate` wie in LaTeX und `\describe` wie `\description`
  - Gruppierung mit der "Liste" mit {<Liste>}; Listenelemente mit `\item`
  - Beispiel:

```
\itemize{
   \item erster Punkt
   \item zweiter Punkt   }
```

  - die Listenbefehle können verschachtelt werden
- Tabellen:
  - Befehl: `\tabular{rcl}` wie in LaTeX d.h. mit jeweils einem Argument aus `r`, `c`, `l` pro Spalte für links-, rechtsbündige bzw. zentrierte Ausrichtung der Tabellenelemente

- Spaltenwechsel mit \tab, Zeilenwechsel mit \cr
- Beispiel

```
\tabular{rlll}{
[,1] \tab Ozone \tab numeric \tab Ozone (ppb)\cr
[,2] \tab Solar \tab numeric \tab Solar R (lang)\cr
}
```

8.2.6 (e) spezielle Befehle à la LaTeX zum Setzen von Gleichungen

- \eqn[{<latexcode>}]{<asciicode>} für Formeln innerhalb des Texts
- \deqn[{<latexcode>}]{<asciicode>} für abgesetzte Formeln

wobei <latexcode> LaTeX-code im Mathematikmodus ist und <asciicode> bei HTML, chm und R-Hilfe-Format verwendet wird

8.2.6 (f) Sonderzeichen

- R $\hat{=}$ ("wird geschrieben als") `\R`,
- ... als Funktionsargument $\hat{=}$ `\dots`
- ... als Ellipse im laufenden Text $\hat{=}$ `\ldots`
- % ist ein Kommentarzeichen; Text, der in dieser Zeile folgt, wird ignoriert
- \, %, {, } werden generell maskiert
- innerhalb von code-artigen Umgebungen wie `\code`, `\preformatted`, `\examples` (aber nicht `\file`) müssen daneben keine weiteren Zeichen maskiert werden
- im laufenden Text müssen daneben auch '$', '#' und '_' maskiert werden
- '^' $\hat{=}$ `\eqn{\mbox{\textasciicircum}}{^}`,
  '~' $\hat{=}$ `\eqn{\mbox{\textasciitilde}}{~}`,
  '~' $\hat{=}$ `\eqn{\sim}{~}`,

- '<', '>', and '|' stehen nur in \eqn bzw. \deqn zur Verfügung
- ä, Ä, ö,Ö, ü,Ü, ß, œ, Œ, æ, Æ, å, Å, ø, Ø, ł, Ł, ¿, ¡, sowie ó, ò, ô, õ, ȯ, ŏ, ǒ, ő, o͡o, o̧, o̩, o̲ und ähnliche Sonderzeichen können mithilfe von Konstruktionen wie \enc{¨o}{o} eingefügt werden, wobei das erste Argument verwendet wird, wo andere Zeichensätze erlaubt sind, und das zweite eine ASCII-Transkription ist

8.2.6 (g) Format der Hilfe-Files

Dokumentation kann automatisch in verschiedenen Formaten wie HTML, ASCII, Nroff–Format, dvi und PDF erzeugt bzw. umgewandelt werden — Befehle

- R CMD Rdconv konvertiert von einem R-Dokumentationsformat in ein anderes oder extrahiert die lauffähigen Beispiele zum online-Testen; Formate ASCII , HTML, LATEX, S3– und S4–Dokumentationsformat

- **R** CMD Rd2txt erzeugt "schöne" formatierten Text
- **R** CMD Rd2dvi erzeugt `DVI` oder, mit Option `--pdf`, `PDF` Format
- **R** CMD Sd2Rd erzeugt `S4` Doku-Format (nutzt `SGML`-Format) aus `S3` Doku-Format (nutzt `Nroff`-Format) aus

8.2.6 (h) Dokumentation von `S4`-Klassen und -Methoden

[experimentell, noch im Umbruch]

- jede nach außen sichtbare Klasse und Methode sollte zumindest einen \ **alias** -Eintrag haben
- **wichtig:** \ **alias** -Einträge sind von der Form `<clsname>-class` bzw. `<genFkt>,<signatureList>-method`, wobei `<signatureList>` eine Liste mit der Signatur der Methode ist, kommagetrennt und ohne Freizeichen zwischen den Einträgen der Signatur (sonst funktioniert der **help**-Mechanismus für `S4`-Klassen nicht richtig)
- Hüllen werden mit promptMethods und promptClasses angelegt

- zusätzlich, falls eine spezielle Methode besonders dokumentiert werden soll:
  \S4method{<genFkt>}{<signatureList>}(<argumentList>),
  \S3method{<genFkt>}{<signatureList>}(<argumentList>)

8.2.6 (i) Dokumentation des Pakets an sich

- Pakete können eine Übersichts-Hilfeseite erhalten — mit einem \alias{<pkgname>-package},
- dann: package?<pkgname> öffnet diese Hilfeseite; noch besser (falls kollisionsfrei möglich) zusätzlich \alias{<pkgname>}
- Schablone mit promptPackage()
- Anordnung des Inhalts ist weitestgehend freigestellt; obligatorischer Tag: \docType{package}
- Empfehlung: kurze Übersicht; etwas detaillierte Dokumentation sollte in eine Vignette

Erinnerung: Außerhalb von Paketen werden einzelne Routinen mit Hilfe–Files versehen wie in Abschnitt 3.3.4 beschrieben

8.2.6 (j) Vignetten

[nach einem DSC–Tutorial 2003 von D. Bates und T. Lumley]

- Idee: Literate Programming:
  - http://www.literateprogramming.com/
  - http://en.wikipedia.org/wiki/Literate_programming

  $\rightsquigarrow$ Dokumentation und Quelltext werden
  - in *ein* File geschrieben und
  - anschließend mit speziellen Werkzeugen verarbeitet
- `Sweave`:
  - Entwickler: Friedrich Leisch
  - Idee: Verbinden von Dokumentation / Artikel und lauffähigem R–Code $\rightsquigarrow$ *reproducible research*
  - man schreibt einen leicht erweiterten LaTeX–Code — Endung des Files `.Snw` bzw. `.Rnw`

- `.S[/R]nw`-File wird mit mit dem Befehl Sweave in `R` übersetzt
- Sweave extrahiert den Code, lässt ihn unter R ablaufen und setzt den Output wieder ins Dokument ein
  + Leser sehen, dass der Code funktioniert
  + Input, Output und Text können beliebig angeordnet werden
  + das Dokument kann erneut unter R abgearbeitet werden, um Resultate aufzudatieren
- Auslösen von R-Code aus `.S[R]nw`-File in R mit Stangle
- aktuelle Dokumentation:
  `http://www.ci.tuwien.ac.at/~leisch/Sweave`
- Alternative zu `Sweave` jenseits von TEX:
  * Verweben mit `ODF` (Open Document Format) anstelle von LATEX-Code
  * in R mit Paket `odfWeave`

- Vignetten:
  - Idee: Lücke zwischen Funktionsdokumentation und Büchern über `R` zu stopfen
  - stark verwendet im `Bioconductor Project`
  - was in eine Vignette steht:
    + ein einzelnes Thema
    + lauffähiger Code basierend auf Daten, die in `R` verfügbar sind; die benutzten Libraries müssen dokumentiert sein
    + sollte mehr als eine einzelne Funktion sein; sollte einen Prozess / eine Problemlösung beschreiben und typischerweise mehrere Funktionen umfassen
  - Aufruf von Vignetten von R aus mit vignette()
  - Referenz: Writing R Extensions (2006b), Kapitel 1.4

R–Beispiel 8.2-2 [Vignetten]:

```
## Auflisten aller vorhandenen Vignetten
vignette()
## Aufruf Vignette zu Paket grid
vignette("grid")
#oder
v1 ← vignette("grid"); print(v1)
## Herausloesen des Codes
edit(v1)
```

8.2.6 (k) Qualitätskontrolle

- Das `packaging system` kontrolliert die Dokumentation, ob
  - alle Objekte dokumentiert sind (d.h. ein entsprechender \alias-Eintrag vorliegt)
  - die Dokumentation mit dem dokumentierten R-code konsistent ist

- die Angabe unter \usage konsistent zur Definition der Funktion ist
- die Beispiele in der Sektion \examples laufen

$\rightsquigarrow$ Mindeststandard für die Dokumentation

8.2.7 Namespaces

- Problematik: viele Entwickler schreiben simultan an ihren Routinen $\rightsquigarrow$ früher oder später kommt es zu Konflikten mit Namen von Objekten in verschiedenen Paketen
- Ausweg: *Namespaces* —vergleiche Tierney (2003a), Ligges (2005, S. 81ff und Abschnitt 10.6), sowie
  http://en.wikipedia.org/wiki/Namespaces
- Ideen:
  - nur noch bestimmte Objekte eines Paketes sind für den Nutzer sichtbar; deren Namen werden explizit *exportiert*
  - alle anderen Objekte sind nur für andere Funktionen im selben *Namespace* sichtbar
  - in eigene Pakete (mit eigenem Namespace) kann man auch Objekte aus anderen Paketen *importieren*; deren Namen kann dann nicht mehr vergeben werden

- Kommt es dennoch zu Namenskollisionen,
  - überschreibt das entsprechende Objekt `<name>` aus dem eigenen Namespace nicht dasjenige `<name>` aus einem fremden, sondern *maskiert* es nur;
  - das Objekt `<name>` aus dem fremden Namespace bleibt erhalten, ist nur nicht mehr mit `<name>` ansprechbar;
  - auf exportierte Objekte aus Paket `<pkg>` greift man mit `<pkg>::<objektname>` zu —dies funktioniert auch bei Maskierung!
  - auf nicht exportierte Objekte aus Paket `<pkg>` greift man mit `<pkg>:::<objektname>` zu —dies funktioniert auch bei Maskierung!
  - Achtung: Zugriffe mit `::` oder gar `:::` sind aufwendiger; es wird das entsprechende Paket `<pkg>` geladen aber nicht in den Suchpfad eingetragen!

- nützlich zum Suchen von Objekten:
  getAnywhere: sucht nach Objekten im Suchpfad und in geladenen Namespaces, auch wenn das Objekt nicht exportiert oder der Namespace in den Suchpfad gehängt ist
- Anlage eines Namespace:
  - im Pakethauptverzeichnis legt man eine (Text-)Datei `NAMESPACE` an
  - Befehle
    * `export(<obj1>,<obj2>,....)`, `exportPattern(<reg.Ausdruck>)` (vgl. 1.10-1), exportieren (klassische) Objekte
    * `import(<pkgname>)`, `importFrom(<pkgname>,<obj1>,<obj2>,....)` importieren alle / bzw. die gelisteten (klassischen) Objekte aus Paket `<pkgname>`
    * für S3-Klassen: `S3method(<mtdname>,<clsname>)` registriert die S3-methode `<mtdname>.<clsname>()`

    - für S4-Klassen:
        - per .onLoad←**function**(<lib>,<pkg>) **require**(**methods**) zu Beginn des ersten eingeladenen source-files muss das methods-Paket eingeladen werden
        - alle nach außen sichtbaren S4-Klassen und generischen Funktionen müssen mit exportClasses(), exportMethods(), exportiert werden
        - alle S4-Klassen und generischen Funktionen aus anderen Paketen, die im eigenen Paket genutzt werden, müssen per importMethodsFrom(), importClassesFrom() importiert werden
    - für externen Code: useDynlib(<dllname>)
- löschen eines Namespaces mit
- alles noch im "Werden" $\rightsquigarrow$ Änderungen wahrscheinlich

8.2.8 Vorbereiten der Anlage eines Pakets

- die Funktion **package**.skeleton automatisiert teilweise das Anlegen eines Pakets mit einer korrekten Struktur und Dokumentation
- Syntax:
  package.skeleton(name="<pckname>",list=c("<inh1>", "<inh2>",....),environment=.GlobalEnv, path=".", force=FALSE)
- legt ein R Paket "<pckname>" im Verzeichnis unter Argument path an
- in dieses Paket kommen die Objekte aus list oder aus der Umgebung environment
- dabei werden die Objekte aufgeteilt in Datensätze $\rightsquigarrow$ data/– und Funktionen $\rightsquigarrow$ R/–Verzeichnis
- Skelette für die Help–Files für klassische Objekte werden mit

dem **prompt**–Befehl erzeugt, für S4-Klassen und -Methoden mit promptClasses, promptMethods

- ein DESCRIPTION–File wird angelegt
- dann gibt die Funktion eine Liste an zu erledigenden Dingen aus

8.2.9 Anlage eines Pakets

- mithilfe des Befehls **R** CMD build unter UNIX bzw. Rcmd build unter Windows
- das Resultat kann dann einfach von einem System zum anderen transportiert und ohne Entpacken installiert werden
- es gibt Optionen, Help– und Daten–Files in permanent gepackter Form zu speichern (nützlich auf alten Windows–Systemen, wo viele kleine Files viel Festplattenspeicher schlucken)

---

hier gibt es definitiv Schwierigkeiten bei Win 9x unde ME;
ab R 1.9.0 ist auch unter Windows die **R** CMD <Befehl>-Notation möglich.
Unter Windows sind aber Vorabeiten nötig; siehe Abschnitt 8.2.12

- um Windows-Binaries zu erzeugen: **R** CMD build −−binary
- hierfür braucht man `Perl`; dieses muss im Windowspfad stehen
- in Win 9x/ME Schwierigkeiten mit langen Filenamen!

8.2.10 Binär– und Quell–Pakete

- **R** CMD build erstellt Pakete aus R–Quellfiles
- wenn `C`– oder `FORTRAN`–Routinen mit benötigt werden, sollten für Windows Nutzer auch Binärdateien dieser Routinen zur Verfügung gestellt werden — siehe auch `R for Windows FAQ`
- Binärpakete werden mit **R** CMD build −−binary angelegt;
- die meisten R–Entwickler arbeiten unter `Unix`; selbst Windows–lauffähige Binarys werden unter `Unix` mit *Crosscompilern* erzeugt (vgl. R Installation and Administration (2006d, Abschnitt 3.1.8))

8.2.11 Checken eines Pakets

- mithilfe des Befehls **R** CMD **check** unter `UNIX` bzw. Rcmd **check** unter `Windows`
- unterstützt Qualitätsanalyse und –kontrolle (QA/QC) für Pakete
- die Verzeichnisstruktur und das Format des `DESCRIPTION` und `INDEX`–Files werden geprüft
- die Dokumentation wird in `text` in `HTML` und LaTeX gewandelt
- wenn LaTeX auf dem System gefunden wird, wird die Dokumentation geTeXt
- die Beispiele werden ausgeführt
- alle Tests im `tests/`–Verzeichnis werden ausgeführt

---

hier gibt es definitiv Schwierigkeiten bei Windows 9x ME;
ab `R 1.9.0` ist auch unter Windows die **R** CMD `<Befehl>`-Notation möglich.
Unter Windows sind aber Vorabeiten nötig; siehe Abschnitt 8.2.12

- undokumentierte Objekte und solche, bei denen usage und Definition nicht übereinstimmen, werden gemeldet

8.2.12 Vorbereitungen zur Erzeugung von R-Paketen unter Windows

dieser Abschnitt wurde im wesentlichen von *Matthias Kohl* beigetragen

- Referenz: http://www.murdoch-sutherland.com/Rtools/

1. Installation von: Tcl, Perl, MinGW (evtl. MS HTML Compiler)
   - http://www.activestate.com/Products/ActiveTcl/Download.html
   - http://www.activestate.com/Products/ActivePerl/Download.html
   - http://www.mingw.org/download.shtml
   - http://www.microsoft.com/downloads/details.aspx?FamilyID=
     =00535334-c8a6-452f-9aa0-d597d16580cc&DisplayLang=en

   man benötigt `MinGW` mindestens in Version `MinGW-5.0.2.exe` und (z.Z.) zusätzlich mingw-runtime $>= 3.10$, was man am

besten direkt erhält unter

http://prdownloads.sourceforge.net/mingw/mingw-runtime-3.10.tar.gz?download;

frühere Versionen gehen nicht;

MS HTML Compiler ist evtl. bereits installiert

2. Für Unicode-Unterstützung unter WinME, Win9X: (ab R 2.1.0) benötigt:
   - Opencow library: opencow.dll, installiert in im R–Stammverzeichnis unter src/gnuwin32/unicode
   - libunicows import library: libunicows.a, installiert im Verzeichnis MinGW/lib directory
   - iconv internationalization conversion library: iconv.dll, installiert im Verzeichnis src/gnuwin32/unicode
   - weitere Details: im File src/gnuwin32/unicode/INSTALL (unter dem R–Stammverzeichnis)
3. RTools ins R\tools Verzeichnis entpacken; zu beziehen von

   http://www.murdoch-sutherland.com/Rtools/tools.zip

4. beim Umgang mit mehreren Versionen von R simultan sehr nützlich: die batch-utilities von Gabor Grothendiek, zu entpacken am besten in `R\batch-utils` Verzeichnis; Details siehe `README`-Datei im entsprechenden `.zip`-File
   - http://cran.r-project.org/contrib/extra/batchfiles/
5. Pfad: Ergänzen von `Tcl`, `Perl`, `MinGw`, `HTML Compiler`, `Rtools`, und R Version bzw. `batch-utils`
   - Bsp: (mit batch-utils)

     ```
     .;C:\Programme\R\batch-utils;C:\Programme\R\tools;
     C:\Tcl\bin;C:\texmf\miktex\bin;C:\Perl\bin\;
     C:\Mingw\bin;C:\Programme\HTML Help Workshop
     ```

   - unter `XP`: Pfad zugänglich unter `Systemsteuerung` → `System` → `Erweitert` → `Umgebungsvariablen`
   - auf kurze Pfadnamen achten —sonst schneidet Windows mitten im Pfadnamen ab...!

6. Anpassen der Datei `MKRules`: (im Verzeichnis `/src/gnuwin32`) (Achtung: möglichst keine Leerzeichen im Pfad, da es sonst evtl. nicht funktioniert ...)
   - `HHWDIR="C:/Programme/HTML Help Workshop"` (Pfad für MS HTML Compiler)
   - `TCL_HOME = C:/Tcl` (Pfad für Tcl)
   - `HEADER= C:/MinGw/include` (Pfad für MinGw)
7. evtl. Anlage einer Umgebungsvariable `TMPDIR`
8. bei Nutzung von `MikTeX >=2.4`:
   Verändern der (lokalen [!]) `miktex.ini`-Datei: in folgenden Rubriken jeweils den Pfad auf die aktuelle Version des files `rd.sty`, im R Stammverzeichnis unter `share\texmf`
   —vgl. http://www.murdoch-sutherland.com/Rtools/miktex.html
   - `[LaTeX]`, `[eLaTeX]`, `[TeXinfo]`, `[pdfteX]`,
   - `[pdfetex]`, `[pdfLaTeX]`, `[pdfeLaTeX]`, `[pdftexinfo]`

Bemerkung 8.2-3 [R selbst compilieren]:

- nach diesen Vorbereitungen: R auch auf eigenem Rechner aus Sourcen compiliebar
- interessant für Paketentwickler, wenn eine neue R-Version kurz vor Veröffentlichung und zur Devel-Version nicht ständig neue Windows-Installer gebaut werden
- Details hierzu R Installation and Administration (2006d, Abschnitt 3.1 und Appendix F)

8.2.13 Erstellen von Bundles

- manchmal nützlich: Distribution mehrerer Pakete im Verbund als ein *Bundle*
- Beispiele: `VR`, `RobASt`
- Installation: wie ein Paket
- Deinstallation: keine (Stand `2.0.1`) Deinsallation als Verbund sondern jedes Paket einzeln
- Erstellung: wie bisher;
- `DESCRIPTION` files sehen anders aus:
  - das "Haupt"-`DESCRIPTION` file sieht en gros aus wie bisher
  - nicht enthalten: Punkte `Package`, `Description`
  - neu dabei: Punkte
    * `Bundle`: (Bundle-Name)
    * `BundleDescription` Beschreibung des Bundles

    * `Contains`: Liste der in dem die verbundenen Pakete (durch Freizeichen getrennt)

- `DESCRIPTION` files der eigentlichen Pakete:
  - heißen `DESCRIPTION.in`
  - enthalten nur Punkte `Package` und `Description`

8.2.14 Weitergabe eines Pakets/Bundles

siehe Abschnitt 8.4

8.2.15 Erfahrungen mit dem Schnüren von Paketen

dieser Abschnitt wurde von *Matthias Kohl* und *Thomas Stabla* beigetragen

- **prompt** promptClasses promptMethods
  - für Dokumentation von Funktion und Variablen ist **prompt** vorgesehen (auch für generische Funktionen!)

- promptClasses zur Dokumentation von S4 Klassen
- promptMethods zur Dokumentation von S4 Methoden
- Unterschiede
  - Im GGs. zu **prompt** und promptClasses: bei promptMethods keine Meldung der Form "Created File ..."
- empfohlene Art der Dokumentation von Methoden
  - bei generischer Funktion $\rightsquigarrow$ "\S3method" "\S4method"
  - bei Klassen: S4 Methoden (z.B. Accessor-/Replacement Funktionen)
  - Dokumentation bei Klassen problemlos, wenn Methode nur mit ihrer Signatur aufgerufen wird, nicht mit weiteren Parametern
- von promptMethods erzeugte Rd-Datei kann verwendet werden
  - zur Dokumentation von Methoden —vgl. ?Documentation
  - zur Kopie der " alias "–Zeilen und des "Methods"–Abschnitt in bereits existierende Hilfedatei

- Wie erstellt man am besten ein `bundle`
  - Erzeugung im Wesentlichen wie bei Paket
  - es gibt ein `DESCRIPTION` file für das Bundle

dort im Feld `Contains` anzugeben: Welche Pakete umfasst das Bundle?

  - Reihenfolge relevant!! d.h., falls Paket des Bundles anderes Paket des Bundles benötigt, $\rightsquigarrow$ erstere nach dem zweiten im `Contains`-Feld
  - für die eingeschlossenen Pakete gibt es keine einzelnen `DESCRIPTION` files, sondern `DESCRIPTION.in` files
  - `DESCRIPTION.in` file kann im Prinzip alle Punkte enthalten, die nicht bereits im `DESCRIPTION` file des bundles enthalten sind — insbesondere `Title`, `Depends`, `Suggests`, `Imports`
  - zusätzlich: Feld `BundleDescription`
  - bei Abhängigkeiten innerhalb des Bundels möglicherweise notwendig:

    - Abhhängigkeiten mit `.onLoad` Funktion zu "erfüllen"
    - Aufnahme im `Depends`-Feld der `Description.in` Datei reicht nicht! (Fehlermeldung deutete auf Zusammenhang mit den Namespaces)

- Pakete schnüren unter Linux:
  - beim Installieren von / Arbeiten mit R unter Linux:
    - in SuSE Linux Distribution, Version 9.2 ist R nicht enthalten,
    - $\rightsquigarrow$ vorkompiliertes `rpm`-Paket vom CRAN laden oder Sourcen holen und selbst kompilieren...
    - im zweiten Fall: notwendig sind auf jeden Fall ein `C` und ein `Fortran` Compiler
    - typischer Fehler: vergessen, Linux-Pakete `readline`, `readline-devel`, `xorg-x11-devel` zu installieren
    - erste beide notwendig für "History editing"-Funktion
    - dritte nötig, um überhaupt Graphiken am Bildschirm anzeigen lassen zu können

- beim Erstellen von Paketen:
  - hilfreich Ausgabe des Kommandos:

    R CMD <command> −−**help** — v.A.

    <command> = build | **check**
  - dabei <command> aus der commands-Liste aus dem Aufruf

    R −−**help**
  - Beispiel: Ausgabe von `R CMD check -help`

```
Usage: R CMD check [options] pkgdirs

Check R packages from package sources in the directories
specified by pkgdirs.  A variety of diagnostic checks on
directory structure, index and control files are performed.
The package is installed into the log directory (which
includes the translation of all Rd files into several formats),
and the Rd files are tested by LaTeX (if available).  All
examples and tests provided by the package are tested to see
if they run successfully.

Options:
 -h, --help          print short help message and exit
 -v, --version       print 'check' version info and exit
 -l, --library=LIB   library directory used for test
                     installation of packages (default is outdir)
 -o, --outdir=DIR    directory used for logfiles, R output, etc.
                     (default is 'pkg.Rcheck' in current directory,
```


```
                  where 'pkg' is the name of the package checked)
  --no-clean      do not clean outdir before using it
  --no-codoc      do not check for code/documentation mismatches
  --no-examples   do not run the examples in the Rd files
  --no-install    skip installation and associated tests
  --no-tests      do not run code in tests subdirectory
  --no-vignettes  do not check vignettes in Sweave format
  --no-latex      do not run LaTeX on help files
  --use-gct       use 'gctorture(TRUE)' when running examples/
                  tests

By default, all test sections are turned on.

Email bug reports to <r-bugs@r-project.org>.
```


8.3 Schnittstellen zu anderen Programmiersprachen

8.3.1 Wozu ist das gut?

- um Code, der von Experten außerhalb von R geschrieben ist, nutzbar zu machen, insbesondere aus numerischen Bibliotheken wie `Netlib`
- manche (wenige!) Operationen lassen sich schlecht vektorisieren, und hier sind dann compilierte Sprachen überlegen

ABER zuerst versuchen Code zu profilieren — siehe Abschnitt 8.3.2

8.3.2 vor Nutzung von compiliertem Code: Profiling von R–Code

- nicht immer ist eine Portierung zeitkritischer Teilroutinen in C notwendig; alternativ kann man den R–Code erst bereinigen und dann profilieren
- Referenz: Writing R Extensions (2006b, Abschnitt 3)
- erster Schritt: Bereinigen von R–Code
  - R–Code aus einer Library und solcher, der über den Prompt / mit Source eingelesen werden in R unterschiedlich behandelt:
    - bei benutzer–eingegebenem Code wird der Source–Code immer mitgeführt und jedes Mal neu geparst
    - bei Code aus Libraries wird nur noch der Parsing–Baum eingelesen
    - "ent–Parsing", i.e. automatisches Regenerieren des Source–Codes aus dem Parsing–Baum kann einen

konsistenten, leichter lesbaren Code liefern, i.e.

- einheitliche Einrücktiefe
- einheitliche Konvention beim Zeilenumbruch vor / nach Funktionsdeklarationen
- einheitliches Setzen von Spacings vor und nach Operatoren
- Verwendung eines einheitlichen Zuweisungsoperators

* dazu muss das Source–Attribut der entspr. Funktion gelöscht werden
  - indem man die Option keep.**source** auf FALSE setzt
  - indem man das Attribut explizit löscht:

    attr(myfun,"source")←NULL
* dann alle "ent–parsten" Funktionen zur Kommentierung und Nacheditierung in ein File **dump**en:

```
options(keep.source=F)
source("myfuns.R")
dump(ls(all=TRUE),file="new.myfuns.R")
```


- zweiter Schritt: Profiling von R–Code
  - Profiling ist für Unix und Windows verfügbar, aber nicht für Macintosh!
  - Befehl: Rprof — Details siehe **help**(Rprof)
  - für Rcmd Rprof muss unter Windows `Perl` installiert sein
  - Idee: Rprof nimmt in festen Zeitabständen (empfohlen: 10ms auf einem 1GHz Rechner), welche R–Funktion gerade benutzt wird und schreibt das in ein File — per default `Rprof.out` im Arbeitsverzeichnis
  - anschließend kann man sich mit summaryRprof im prompt oder mit `R CMD Rprof Rprof.out` [Unix] bzw. `Rcmd Rprof Rprof.out`[Win] die Aktivitäten zusammenfassen lassen
  - ersteres erzeugt ein R–Objekt, braucht kein `Perl` ist aber langsamer (z.B. 4 mal so langsam im Beispiel in Writing R Extensions) als letzteres

  $\rightsquigarrow$ Identifikation der Engpässe — nicht immer braucht man

dann zu deren Behebung compilierten Code

- dritter Schritt: Verwendung von Compiliertechniken für R–Code
  - steckt noch in den Kinderschuhen
  - Positionspapier von Tierney (2003b)

    http://www.stat.uiowa.edu/~luke/R/bytecode.html
  - folgende Features sollten (automatisch) umgesetzt werden
    * Herausnahme unsinniger Dispatching Tests
    * Reduktion der "Lookup"–Kosten durch Beschränkung auf lokale Symbole
    * versiegelte (i.e. nicht zum Überschreiben freigegebene) Symbole als solche erkennen
    * Ersatz "konstanter" Ausdrücke durch ihren Wert
    * schließlich ein Byte–Code Compilat der Source

8.3.3 Schnittstellen von und zu anderen Programmiersprachen

- C / FORTRAN und R
  siehe Abschnitt 8.3.5
- JAVA und R
  - S-JAVA
- XLisp und R
  - RXLisp
- Python und R
  - RSPython
- Perl und R
  - RSPerl
- MATLAB und R
  - RMatlab

8.3.4 Schnittstellen zu Datenbanken — *R* und MySQL

8.3.4 (a) Motivation

- R und Datenbanken verfolgen subtil unterschiedliche Zielsetzungen (vgl. Abschnitt 0.2.3)
- dennoch hilfreich bei sehr großen Datensätzen:
  - klassisches Filesystem stößt an Grenzen
  - Nutzung von SQL zum Datenzugriff
- vergleiche auch Abschnitt 4 in "R Data Import/Export"
- gemeinsames "Frontend"-Paket; `DBI`, verschiedene Datenbanktreiber im Backend — hier nur Schnittstelle R und MySQL $\rightsquigarrow$ `RMySQL`

- weitere Schnittstellen zu `Oracle` (Paket: `ROracle`) und `SQLite` (Paket: `RSQlite`)

8.3.4 (b) Installation

- unter `Linux`; standard (`R CMD install RMySQL`); etwas umständlicher unter Windows (im folgenden):
- Schritt 1: Installation des DBMS (Data Base Management System) — Download unter http://www.mysql.com/
- Schritt 2: Setzen von Pfaden (in Windows) z.B. **path**=%**path**%;**C**:\PROGRA~1\MySQL\MYSQLS~1.0\lib\opt
- Schritt 3: in R: Installation von Paket `DBI`
- Schritt 4: Vorbereiten des PC zum Erstellen von R Paketen (Tcl/TK, Perl, MinGw, ...), vgl. Abschnitt 8.2.12

- Schritt 5:folgende Answeisungen an die Kommandozeile

```
cd C:\PROGRA~1\MySQL\MYSQLS~1.0\lib\opt
REM ## oder wo sonst MySQL liegt

REM erzeuge LIBMYSQL.def
reimp --only-def libmySQL.lib

REM erzeuge libmySQL.a (eine Zeile!)
dlltool --dllname libmySQL.dll --def LIBMYSQL.def
        --output-lib libmySQL.a  -k

REM eigentliche Installation
R CMD install RMySQL
```

- in R: Einladen wie gewohnt mit **require**(RMySQL)
- für R-2.3.1 und MySQL Server 5.0: hier anstelle von Schritt 4 und 5: RMySQL.zip

- hier nicht: Anlage von Datenbanken / Designfragen, i.e. `CREATE TABLE`, `CREATE INDEX`
  (vgl. entsprechende Informatik-Vorlesungen; oder kurz: Wikipedia)
- hier nicht: Datenmanipulation: i.e. `INSERT`, `UPDATE`, `DELETE`
- wichtigstes Konstrukt für uns: die Abfrage, i.e. `SELECT` Struktur:

```
SELECT [DISTINCT] Auswahlliste
       FROM Quelle
       WHERE Where-Klausel
       [GROUP BY (Group-by-Attribut)+
              [HAVING Having-Klausel]]
       [ORDER BY (Sortierungsattribut)+
              [ASC|DESC]]
```

8.3.4 (d) Hauptbefehle in DBI

- Verbindung mit einer bestehenden Datenbank
  - Initialisierung eines mySQL-Clients:
    m ← MySQL() oder m ← dbDriver("MySQL")
  - Nutzer-Identifikation: geht zwar auch mittels dbConnect; aber aus Sicherheitsgründen besser: .my.cnf file, vgl. Seite 3 im Manual zu RMySQL
  - Verbindung mit Datenbank dbConnect
- Übermittlung einer SQL-Anfrage
  - dbSendQuery (Ergebnis vom Typ "DBIResult")
  - dbGetQuery (Ergebnis vom Typ data frame)
  - anschließend: (z.B. nach fetch mit dbClearResult: löschen der Ergebnisse
- fetch: gezieltes Auswählen einzelner "Zeilen"/Datensätze als Liste

- ein Beispiel:

```
## Not run:
# create an MySQL instance and
#   set 10000 of rows per fetch.
m ← dbDriver("MySQL", fetch.default.records=10000)
con ← dbConnect(m)
rs ← dbSendQuery(con, paste(
"select␣*␣from␣HTTP_ACCESS",
"where␣IP_ADDRESS␣=␣'127.0.0.1'"))
df ← fetch(rs, n = 50)
df2 ← fetch(rs, n = -1)
dbClearResult(rs)
pcon ← dbConnect(p, group = "wireless")
dbListTables(pcon)
## End(Not run)
```


8.3.4 (e) Schnittstelle über ODBC

- ODBC (Open DataBase Connectivity):
  - standardisierte Datenbankschnittstelle auf Basis von SQL.
  - bietet Programmierschnittstelle (API) zur Entwicklung DBMS-unabhängiger Software
  - ursprgl. von Microsoft basisierend auf Call Level Interface
  - heute weitgehend Standard
- zwei Gruppen von Befehlen stehen zur Vefügung:
  - odbc∗-Anweisungen (niedrige Ebene) ermöglichen Zugriff auf entsprechende ODBC Funktionen mit ähnlichem Namen, z.B.:odbcClearError, odbcClose, etc.
  - sql∗-Anweisungen (höhere Ebene) zum Lesen, Schreiben, Manipulieren von Daten zw. Data Frames und SQL-Tabellen, z.B.: sqlFetch, sqlGetResults, sqlQuery
  - Achtung: oft Schwierigkeiten mit Verwendung von Spezialzeichen wie ä,ö,ß ...

8.3.5 C/FORTRAN–Code in R

Beispiele wenn nicht weiter erwähnt aus Writing R Extensions (2006b, Abschnitt 5)

8.3.5 (a) .Fortran

- Ziel: Verwendung einer Lapack–subroutine zur Berechnung der LU–Zerlegung einer Matrix
- der FORTRAN–Code

```
      SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
*     .. Skalare Argumente
      INTEGER                 INFO, LDA, M, N
*     .. Matrix/Array Argumente ..
      INTEGER                 IPIV( * )
      DOUBLE PRECISION        A( LDA, * )
.........
```


- dieser Code liege in compilierter Form als DLL/Shared Library (nicht als . o–Object–File!) bereit und muss bei Bedarf mit **dyn.load** dynamisch ein– und mit **dyn**.unload ausgeladen werden — siehe Abschnitt 8.3.6
- um dgetrf von R aus aufzurufen: Verwendung von `.Fortran`

R–Beispiel 8.3-1 [FORTRAN in R]:

```
mm← matrix(rnorm(16),nr=4)
mmdc← .Fortran("dgetrf", m=as.integer(4),
          n=as.integer(4), a=as.double(mm),
          lda=as.integer(4), ipiv=integer(4)
          info=integer(1))
str(mmdc)
```

- der Fortran Code muss dabei eine Subroutine sein

- wichtig: Sicherheits–/Absturzrisiko!
  - alle übergebenen Argumente müssen vom exakt richtigen Typ (für die Fortran Routine) sein

  $\rightsquigarrow$ explizites Casting mit **as.integer**,....

  $\rightsquigarrow$ explizite Deklaration der Dimension und des Speichermodus mit den Konstruktoren **integer**(n), **double**(n)....

CAVEAT **as.integer**(4) $\neq$ **integer**(4) —
erstes Objekt hat Länge 1 und Wert 4, zweites gibt einen Vektor der Länge 4 mit 0en

  - der Rückgabewert von **.Fortran** ist eine Liste mit den **gleichen** Elementen wie die Argumente von **.Fortran** — mit entsprechenden Namen so vorhanden
  - eine Tabelle mit den Entsprechungen von R, `FORTRAN` und `C`–Typen findet sich in Writing R Extensions, Abschnitt 4.2

- üblich: "*Wrapper*"–Funktionen, die die Argumente checken, casten und anschließend auch das Resultat prüfen
- R–Beispiel 8.3-2 [FORTRAN–Wrapper]:

```
LUdecom<-function(x){
 x<-as.matrix(x); m<-nrow(x); n<-ncol(x)
 storage.mode(x)<-"double"
 ## x bleibt Matrix nur Speichertyp aendert sich
 dc<-.Fortran("dgetrf", m=m, n=n, a=x, lda=m,
        ipiv=integer(min(n,m)), info=integer(1))
 if(dc$info)
     stop(paste("dgetrf returned error code",
     dc$info))
 list(lu=dc$a, ipiv=dc$ipiv)
}
##Aufrufbeispiel:
mmdc<-LUdecom(mm); str(mmdc)
```


- Erläuterung:
  - **storage.mode**(x)="double" is ähnlich zu **as.double**(x), wirft aber die Attribute von `x` nicht weg
  - R–Matrizen haben als erste Dimension (c.f. `lda`) die Zeilenzahl

8.3.5 (b) .C

- mit der Funktion **.C** ruft man eine C Funktion auf, die als Rückgabewert **void** hat
- **alle** Argumente der C–Funktionen müssen Zeiger ["call by reference"!] sein, genauer vom Typ **int** $*$, vom Typ **double** $*$ oder (seltener) Rcomplex $*$, **char** $**$
- der Rückgabewert / das "output"–Objekt muss in R erzeugt werden und als Argument im Aufruf von **.C** übergeben werden
- Beispiel aus Writing R Extensions — detailliertere Information siehe dort

R–Beispiel 8.3-3 [Faltung zweier Vektoren mit .C]:

– der C Code

```
void convolve(double *a, int *na, double *b,
              int *nb, double *ab)
{
  int i,j, nab= *na + *nb -1;

  for(i=0; i<nab; i++)
     ab[i]=0.0;
  for(i=0; i<*na; i++)
      for(j=0; j<*nb; j++)
          ab[i+j]+= a[i] * b[j];
}
```

– dieser Code liege in compilierter Form fertig als DLL/Shared Library (nicht als . o–Object–File!) bereit und muss bei Bedarf mit **dyn.load** dynamisch ein– und mit **dyn**.unload ausgeladen werden — siehe Abschnitt 8.3.6

– um convolve von R aus aufzurufen: Verwendung von **.C**

```
conv <- function(a,b){.C("convolve",as.double(a),
        as.integer(length(a)),as.double(b),
        as.integer(length(b)),
        ab=double(length(a)+length(b)-1))$ab}
```

- Aufruf von C Code mit .C geht gut mit rein numerischen Berechnungen;
- wird schwerfällig bei Übergabe von komplizierter strukturierten Objekte
- der Großteil von R ist in C geschrieben $\Rightarrow$ interne Darstellung von R–Objekten ist dem C–Programmierer zugänglich

8.3.5 (c) .Call

- bei strukturierten Objekten besser als .C: .Call — sehr mächtiges Tool!

- `.Call` übergibt R–Objekte im Typ raw und gibt auch ein solches R–Objekt vom Typ raw zurück
- wichtig: Sicherheits–/Absturzrisiko! — typisch: "`Segmentation fault`"
  - R hat eine eigene Speicherverwaltung, die eine `Garbage Collection` bereitstellt; diese konfligiert unter Umständen mit der `C`–Speicherverwaltung — siehe Abschnitt 8.3.9

  $\rightsquigarrow$ bei der Erzeugung von R–Objekten in `C` müssen diese vor der `Garbage Collection` mit PROTECT geschützt werden
  - vor der Rückkehr zu R muss die gleiche Anzahl an Zeigern mit UNPROTECT wieder freigegeben werden, die zuvor mit PROTECT geschützt worden ist

R–Beispiel 8.3-4 [Faltung zweier Vektoren mit .Call]:

```
#include <Rdefines.h>
SEXP convolve2(SEXP a, SEXP b) /* SEXP Typ: S-Expression*/
{int i,j, na,nb, nab; double *xa, *xb, *xab; SEXP ab;

 PROTECT(a = AS_NUMERIC(a)); PROTECT(b = AS_NUMERIC(b));

 na=LENGTH(a);   nb=LENGTH(b);   nab=na+nb-1;

 PROTECT(ab = NEW_NUMERIC(nab));

 xa=NUMERIC_POINTER(a); xb=NUMERIC_POINTER(b);
 xab=NUMERIC_POINTER(ab);

 for(i=0; i<nab; i++) xab[i]=0.0;
  for(i=0; i<na; i++) for(j=0; j<nb; j++)
            xab[i+j]+= xa[i] * xb[j];

 UNPROTECT(3); return(ab);}
```


- der R–Code wird nun viel besser lesbar
- alle Argumente der C–Funktion sind nun vom Typ SEXP — ein Zeiger auf eine `symbolic expression`, in Anlehnung an `LISP`
- eine genauere Übersicht über die exakten R–SEXP–Typen findet sich in Writing R Extensions (2006b, Abschnitt 5.7.3)
- diese sind wichtig zum Anlegen von R–Objekten in `C` mit den Makros NEW_xxxx
- Typüberprüfung von R–Objekten in `C` erfolgt mit den Makros isxxxx; welche genau zur Verfügung stehen siehe Header–Dateien `Rinternals.h` und `Rdefines.h`
- wenn man den Aufruf einer C–Funktion nicht selbst verwendet, sollte man wie bei **.Fortran** eine Wrapper–Funktion schreiben, die alle Castings und Typ–Checks durchführt
- das Makro AS_NUMERIC ist ähnlich zur R–Funktion **as.double**

- weil dieses Makro möglicherweise eine Kopie des R–Objekts in einem neuen Speichertyp erzeugt, muss das Ergebnis von AS_NUMERIC PROTECTed werden
- ein bisschen "tricky": automatisches Setzen von Attributen (wie **length**, **dim**):
  - mit Funktionen getAttrib , setAttrib
  - Definitionen für die Attribute in C in `Rinternals.h`
  - Shortcuts: namesgets, dimgets, dimnamesgets als Analoga zu **names**←, **dim**←, **dimnames**←
  - Utilities: GetMatrixDimnames, GetArrayDimnames
  - Definition neuer Attribute mit: install
  - Details: Writing R Extensions (2006b, Abschnitt 5.7.4)
- Vorsicht bei Listen:
  - üblicherweise S–Listen und keine LISP–Listen

  ⇝ isNewList und allocVector(VECSXP,n) statt isList , allocList (n)

- Finden und Setzen von Variablen
  - Äquivalent zu **get**(`name,envir=rho`):
    SEXP getVar(SEXP name, SEXP rho)
  - Äquivalent zu **find**(`name,envir=rho`):
    SEXP findVar(SEXP name, SEXP rho); dazu aber nötig:
    Einrichten von `name` in der Symboltabelle mit install
  - Neuzuweisung / Manipulation von Variablen:
    * **void** defineVar(SEXP symbol, SEXP value, SEXP rho) erzeugt eine neue Zuordnung Speicher $\leftrightarrow$ Symboltabelle; entspricht assign(**symbol**, value, envir=rho, **inherits**=FALSE) — aber keine Kopie der Wertbelegung des Objekts
    * **void** setVar(SEXP symbol, SEXP value, SEXP rho) ändert die Speicherbelegung bei einer bestehenden Zuordnung Speicher $\leftrightarrow$ Symboltabelle; entspricht assign(**symbol**, value, envir=rho, **inherits**=TRUE)

8.3.5 (d) Manipulation von R–Instanzen von R–Klassen in C

- besonders nützlich ist die Kombination von S4–Klassen und dem .Call–Interface
- weil die Typen und Namen von Slots in einer Instanz einer S4–Klasse in deren Klassendefinition definiert sind, kann man hier auf Typprüfung in einer Wrapping–Funktion weitgehend verzichten
- zur Interaktion mit Instanzen von S4–Klassen in C stehen die Makros GET_SLOT, SET_SLOT, NEW_OBJECT, MAKE_CLASS bereit
- Referenz zu diesem Thema (auch Quelle der Beispiele): Bates (2003)

- R–Beispiel 8.3-5 [LU Zerlegung mit S4 Klassen]:
  - Definition einer LU Zerlegungsklasse und eines Konstruktors in R

```
setClass("LUdec",representation(a="matrix",
                 pivot= "integer"))
setClass("dmatrix","matrix",validity=
    function(object){
        mode(object)=="numeric"})
setClass("DGEmatrix","dmatrix")
setGeneric("decompose",function(object ,...){
                 standardGeneric("decompose")})
setMethod("decompose","DGEmatrix",
    function(object ,...).Call("La_DGE_dc",object))
```


– der C–Code dazu:

```
#include <Rdefines.h>
SEXP La_DGE_dc(SEXP A)
{SEXP aa=PROTECT(duplicate(A));
 SEXP adims, pivot, val; int m, n, info;
 if(!isMatrix(aa) || !isReal(aa))
     error("A must be a double precision matrix");
 adims= GET_DIM(aa); m=INTEGER(adims)[0];
 n=INTEGER(adims)[1];
 pivot=PROTECT(NEW_INTEGER(m<n ? m : n));

 F77_CALL(degetrf)(&m,&n,REAL(aa),&m,
          INTEGER(pivot),&info);

 check_Lapack_error(info, "dtrtrf")
 val = PROTECT(NEW_OBJECT(MAKE_CLASS("LUdec")));
SET_SLOT(val, install("a"), aa);
SET_SLOT(val, install("pivot"), pivot);
UNPROTECT(3); return(val);}
```


- Kommentare zu diesem Beispiel:
  - der Wert des Arguments wird im Zuge der FORTRAN–Routine modifiziert $\Rightarrow$ er **muss** daher dupliziert werden, bevor er manipuliert werden kann
  - Makros wie GET_DIM und Funktionen wie isMatrix sind nützlich, um R–Eigenschaften dieser Objekte in C zu bekommen
  - das SET_SLOT Makro [und entsprechend GET_SLOT]:
    * werden bei Instanzen von S4–Klassen benutzt
    * das zweite Argument dieser Funktionen muss ein R–Objekt vom Typ name sein
    * die install –Funktion konvertiert einen C–Character–String in ein solches Objekt
    * install muss verwendet werden, geht aber schnell [wg. Hashing]

- R–Beispiel 8.3-6 [LUdec-Objekt und Matrixinverse]:
  - Nutzung eines Objekts aus der LU Zerlegungsklasse zur Berechung der Inversen der Originalmatrix
  - der C–Code dazu:

```
SEXP La_DGE_tri(SEXP LU)
{
 SEXP aa=PROTECT(duplicate(GET_SLOT(LU,
            install("a"))));
 SEXP pivot=(GET_SLOT(LU,install("pivot")));
 SEXP adims=GET_DIM(aa);
 m=INTEGER(adims)[0]; n=INTEGER(adims)[1];
 int info, lwork=-1; double tmp, *work;

 if(m != n)
    error("LU keine LUzerl. einer quadr.Matrix!");
/* beim ersten Aufruf von dgetri bestimmt:
   optimale Groesse des Arbeits-Arrays */
```


```
 F77_CALL(degetri)(&n,REAL(aa),&n,
          INTEGER(pivot),&tmp,&lwork,&info);

 check_Lapack_error(info, "dtrtri");
 lwork= (int) tmp;
 work = Calloc(lwork, double);

/* eigentlicher Aufruf */
 F77_CALL(degetri)(&n,REAL(aa),&n,
          INTEGER(pivot),&tmp,&lwork,&info);

 check_Lapack_error(info, "dtrtri");
 Free(work);
 UNPROTECT(1);
 return(aa);
}
```


- Kommentare zu diesem Beispiel:
  - wir extrahieren die Slots, die wir brauchen von der Instanz
  - wieder duplizieren wir die Matrix, bevor sie manipuliert werden kann
  - beim ersten Aufruf der Lapack–Routine benutzen wir das Argument `lwork=-1`, um dieser zu signalisieren, dass man nur die benötigte Speichergröße ermitteln will; diese wird als erste Komponente der Arguments `work` zurückgegeben
  - anschließend allozieren wir den benötigten Speicher mit Calloc und geben ihn am Ende der Routine mit Free wieder frei
  - die Funktion check_Lapack_error überprüft, ob der Errorcode des Aufrufs der Lapack Routine nicht-0 ist und gibt gegebenenfalls eine informative Fehlermeldung aus:

```
void check_Lapack_error(const int info,
                        const char* name)
{if (info==0) return;
 if (info<0)
    error("Arg. %d der Lapack Fkt %s ist illegal",
          -info,name);
 error("Lapack Fkt %s gibt Fehlercode %d zurueck",
       name,info);}
```

- Behandlung von Missings und speziellen Werten (IEEE)
  - bei **.C**: sofern Argument `NAOK` nicht auf `TRUE` gesetzt: Fehler bei Auftreten von `NA` oder IEEE-Werten wie `-Inf`, `Inf`, `NaN`
  - **.Call**: Makros zum Test auf solche Werte:
    - für **double**: ISNA, ISNAN, ISNA, **R**_FINITE
    - sonst: NA_REAL, NA_INTEGER, NA_LOGICAL, NA_STRING
    - Konstante (zum Setzen von Werten): **R**_NaN, **R**_PosInf, **R**_NegInf, **R**_NaReal,

- Überblick über die in C zur Verfügung stehenden R–Verfahren, insbesondere für numerische Probleme: Writing R Extensions (2006b, Abschnitt 5.9)

8.3.5 (e) Auswertung von R-Ausdrücken in C

- Hauptwerkzeug: SEXP eval(SEXP expr, SEXP rho);

$\hat{=}$ **eval**(expr, envir=rho)

- hier nicht! — c.f. Writing R Extensions, Abschnitt 4.9

8.3.5 (f) Debugging von compiliertem Code

- will `breakpoints` im kompilierten Code setzen
- Vorgehensweise in `Unix`
  1. Debugger aufrufen mit `R -d gdb`
  2. R aufrufen
  3. am R prompt: Einladen der Library mit **dyn.load** oder **library**

4. senden eines Interrupt–Signals (mit `strg-d`), um in den Debug–Modus zu kommen
5. setzen der `breakpoints` in den Code
6. R weiterlaufen lassen mit `signal 0[enter]`

- Vorgehensweise in `Windows`
  1. setzen eines `breakpoints` für `WinMain`:

```
gdb .../bin/Rgui.exe
(gdb) break WinMain
(gdb) run
# [haelt an, nachdem DLL geladen]
(gdb) break R_ReadConsole
(gdb) continue
# [haelt die R Console an]
(gdb) continue
```

  2. am R prompt: Einladen der DLL mit **dyn.load** oder **library**

3. setzen der `breakpoints` in den Code
4. R weiterlaufen lassen mit

```
(gdb) clear R_ReadConsole
(gdb) continue
```

- in `Windows` funktioniert das Signalisieren nicht so gut $\rightsquigarrow$ besser unter Unix entwickeln....
- Beobachtung von R–Objekten beim Debuggen:
  - mit PrintValue(SEXP s) oder R_PV(SEXP s)
  - Details in Writing R Extensions (2006b, Abschnitt 5.10.2)

8.3.6 dynamisches Einladen von Bibliotheken/DLL's

- damit die Funktionen **.Fortran**, **.C**, **.Call** überhaupt auf die Routinen zugreifen können müssen diese als "*shared library*" in den Arbeitspeicher geladen werden; dies geschieht
- über **library** :
  in den meisten Fällen wird man eine Library/DLL mit dem Befehl **library** .dynam in der Funktion . First . lib –Funktion eines Pakets einladen;
  - . First . lib wird aufgerufen, wenn ein Paket mit **library** oder requirePackage eingeladen wird
  - Syntax: . First . lib `(libname, pkgname)`; dabei sind
  - `libname`: ein String mit dem Namen des Library–Verzeichnis, in dem zu suchen ist
  - `pkgname`: ein String mit dem Namen des Pakets

- Rückgabewert: ein Character–Vektor mit den Namen der Pakete, die diesen Befehl in der aktuellen Sitzung verwendet haben, um shared Libraries zu laden
- Zweck von **library** `.dynam`: in der Funktion `.First.lib` Pakete dynamisch einladen;
- Syntax: **library** `.dynam(chname, package = .packages(), lib.loc = NULL, verbose = getOption("verbose"), file.ext, ...)`; dabei sind
- `chname`: Name der shared Library/DLL
- `package`: eine Liste mit Paketen die nach der DLL durchsucht werden sollen
- `lib.loc`: ein Character–Vektor mit den R–Verzeichnisbäumen, die durchsucht werden sollen oder `NULL`, dann werden alle zur Zeit bekannten Libraries durchsucht
- `verbose`: soll das Einladen auf der Konsole signalisiert

werden?

- `file.ext`: File–Endung der zu ladenden Library
- `...`: weitere Argumente für **dyn.load**
- `new`: ein Character–Vektor mit den Namen der Pakete, die bereits shared Libraries geladen haben
- Rückgabewert: ein Character–Vektor mit den Namen der Pakete, die diesen Befehl in der aktuellen Sitzung verwendet haben, um shared Libraries zu laden

- explizit: Ein/Ausladen der Bibliothek mit den Befehlen **dyn.load**/**dyn**.unload
  - Syntax: **dyn.load**(x, local = TRUE, now = TRUE), resp. **dyn**.unload(x);
    dabei ist
  - x Pfad der Library als String; als Entwickler sollte man keine spezifische File–Endung für die Library festlegen, sondern besser eine Konstruktion wie die folgende verwenden:

```
file.path(path1,path2,paste("mylib",
    .Platform$dynlib.ext, sep="")))
```

    in Unixsystemen kann der Pfad absolut sein, relativ zum Arbeitsverzeichnis oder relativ zum Stammverzeichnis
  - local sollen die Symbol–Namen der Funktionen aus der shared Library lokal in ihrer eigenen Symboltabelle verwaltet werden oder in der globalen Symboltabelle? wird unter Windows ignoriert

- `now` sollen alle Symbole unmittelbar aufgelöst werden (und umplatziert) oder erst beim ersten Auftauch eines solchen Symbols in einem Ausdruck / in einem Aufruf; wird unter `Windows` ignoriert
- weitere Informationen im File `readme.packages`

- Utilities zum Überprüfen welche Routinen im Speicher sind
  - **is**.loaded(<symbol>, PACKAGE="");
    - überprüft, ob das entsprechende Symbol bereits durch ein Einladen einer Shared Library zur Verfügung steht
    - <symbol> ist der Symbol–Name der Funktion (in "·"),
    - im PACKAGE–Argument kann spezifiziert werden, wo das Symbol gesucht werden soll
  - **symbol.C**(<name>)/**symbol.For**(<name>): bilden R–Symbolnamen (Strings!) für C/Fortran–Routinen in die entsprechenden Symbolnamen des compilierten Codes der Shared Library ab

- Achtung: wird ein Symbol in mehreren Bibliotheken verwendet, so überschreibt das zuletzt geladene alle vorangegangen Tabelleneinträge; Ausweg über Namespaces, vgl. Abschnitt 8.2.7
- in Zukunft in Erwägung zu ziehen: Schreiben eigener Registrierungsroutinen für DLL's — c.f. Writing R Extensions (2006b, Abschnitt 5.4)

8.3.7 Erfahrungen mit C-Code in R

dieser Abschnitt wurde von *Thomas Stabla* beigetragen

- ein kleines Beispiel zur Effizienz von C-Code im Vergleich zu R-Code: es sollen die Laufzeiten verglichen werden von
  - **for**-Schleifen in R
  - **for**-Schleifen in C
  - der effizienten R-methode **sum**
- C-Code

```c
void sumInC( double *A, int *length, double *total )
{
    int i;
    *total = 0;
    for( i = 0; i < *length; i++ )
    *total += A[i];
}
```

- R-Code

```
SimpleSum ← function(A) {
  TotalSum ← 0
  for( i in 1 : nrow(A) )
    for( j in 1 : ncol(A) )
      TotalSum ← TotalSum + A[i,j]
  TotalSum
}
```


```
#in Unix
dyn.load("SimpleMath.so")
#in Windows
#   dyn.load("SimpleMath.dll")

is.loaded(symbol.C("sumInC"))

CSum <- function(A) {
   .C("sumInC",
      as.double(A),
      as.integer(length(A)),
      TotalSum = double(1))$TotalSum
}

A <- matrix(rnorm(10^3), 10^3, 10^3)
```


```
system.time(SimpleSum(A))[1:3]
    # => For-Schleife sehr langsam
system.time(CSum(A))[1:3]
    # => C-Code schneller als For-Schleife
system.time(sum(A))[1:3]
    # => effizienter R-Code schlaegt C
    #  (bei Aufruf von .C ein gewisser
    #   Overhead)

#in Unix
dyn.unload("SimpleMath.so")
#in Windows
#   dyn.unload("SimpleMath.dll")
```


8.3.8 Erzeugen von Shared Libraries/DLL's

- in `Linux/Unix`
  - mit dem Befehl **R** CMD SHLIB
  - dieser akzeptiert als Argument eine Liste von Files, die entweder als Object–Files (Endung '`.o`') oder als Sourcen von `FORTRAN`, `C`, `C++` (mit Endungen '`.c`', '`.cc`', '`.cpp`', '`.C`' bzw. '`.f`')
  - Spezifikation zusätzlicher Flaggen im File `Makevars` im Compilierverzeichnis
  - weitere Informationen mit **R** CMD SHLIB −−**help**
- in `Windows`
  - Vorbereitungen wie in Abschnitt 8.2.12
  - mit dem Befehl Rcmd SHLIB

hier gibt es definitiv Schwierigkeiten bei Win 9x unde ME;
ab `R 1.9.0` ist auch unter Windows die **R** CMD `<Befehl>`-Notation möglich.

- empfohlen: Einsatz spezifischer `Makevars.win` resp. `Makefile.win` Varianten der entsprechenden Unix/Linux–Varianten
- weitere Informationen im File `README.packages` im R–Stammverzeichnis; diese File ist Schritt für Schritt abzuarbeiten
  * einmal (nach Neuinstallation einer R-version):

```
cd R_HOME\src\gnuwin32
make libR.a libRblas.a
```

- ACHTUNG: Groß- und Kleinschreibung beachten; insbesondere bei C-files Endung `.c` verwenden (sonst wird C++-Compiler verwendet)!

8.3.9 Koordination der Speichermanager

- in R gibt nicht der Benutzer den Speicher frei, sondern von Zeit zu Zeit findet eine *Garbage Collection* statt, bei der dann der

gesamte (oder zumindest Teile des) ungenutzten Speichers wieder freigegeben werden — c.f. Abschnitt 8.3.10

- erzeugt man ein R–Objekt in C, so muss man R explizit mitteilen, dass dieses Objekt noch benötigt wird, und zwar mit PROTECT
- **Achtung:** das Objekt und nicht die Referenz (der Zeiger) wird geschützt; wichtig bei Neuzuweisungen von Werten an ein Objekt!
- der Schutzmechanismus wird mit einer Kellerstruktur/`stack` verwaltet; Freigabe der obersten (letzten) n geschützten Objekte durch UNPROTECT(n)
- bei endgültigem Rücksprung aus der C Routine müssen alle PROTECT's durch UNPROTECT's gematcht werden — sonst Warnung "stack␣imbalance␣in␣.Call"
- während des Ablaufs der C–Routine kann theoretisch jede (für den Nutzer nicht unmittelbar sichtbare) Zuweisung eine Garbage

Collection auslösen,...

$\rightsquigarrow$ im allgemeinen PROTECT verwenden

ABER in manchen Fällen genauere Analyse sinnvoll, insbesondere bei Verwendung großer Objekte — PROTECT/UNPROTECT für mehrere Tausend Objekte auf einmal lässt den `stack` überlaufen!

- in solchen Fällen: am besten die Objekte als Slots/Elemente eines größeren Objekts schützen
- PROTECT ist nicht nötig für Funktionsargumente — hier "weiß" R, dass sie in Benutzung sind
- bei der Speicherverwaltung selbst dann zwei Strategien

8.3.9 (a) R verwaltet den Speicher

- am Ende des Aufrufs von **.C**/**.Call** verfügt R über den allozierten Speicher

- **char*** R_alloc(**long** n, **int** size): alloziert n Einheiten à size bytes
  typische Verwendung:
  x= (**int** *) R_alloc(nrows(merge)+2, **sizeof**(**int**));
- analog:
  - S_alloc: initialisiert den allozierten Speicher mit 0
  - S_realloc(**char*** p, **long** new, **long** old, **int** size) — modifiziert die Speichergröße von `old` auf `new` Einheiten und initialisiert die neuen Einheiten mt 0
- allozierter Speicher wird vom `Heap` genommen

8.3.9 (b) `C` verwaltet den Speicher

- R–Interface zu malloc — kümmert sich um Fehlerbehandlung
- eigene Funktionen zur Allozierung und Freigabe von Speicher:
  - allgemeine Schnittstellenfunktionen — Analoga zu calloc, realloc, free:

  * type* Calloc(size_t n, type)
  * type* Realloc(any *p, size_t n, type)
  * **void** Free(any *p)

  – spezielle allocxxxx–Funktion bereits in `Rinternals.h` vordefiniert; hier Kenntnis der R–Typen in `C` nötig, c.f. Writing R Extensions, Abschnitt 4.7.3

  – spezielle NEW_xxxx–Makros in `Rdefines.h` vordefiniert

- damit alles gut geht: explizites Casting!
  — z.B. von INTEGER auf REAL mit

  – PROTECT(<newSexp>=coerce(<oldSexp>,REALSXP));

  – PROTECT(<newSexp>=AS_NUMERIC(<oldSexp>));

8.3.10 Exkurs: Speichermanagement in R

- in der absoluten Mehrzahl der Fälle kein Benutzereingriff nötig
- Ausnahmen:

- Schnittstellenprogrammierung, c.f. Abschnitt 8.3
- extrem große Objekte
- den dynamischen Speicher–Alloziermechanismus von R daran hindern mehr als erträglich Ressourcen für R abzuziehen

8.3.10 (a) Wie alloziert R den benötigten Speicher?

- beim Aufruf von R bekommt R einen Default Arbeitsspeicher;
- dieser teilt sich auf in Heap und Felder von `cons cells`
  - der Heap wird für Objekte von variabler Größe verwendet und ist als `heap` von "`Vcells`" à $8\,\mathrm{Byte}$ organisiert
  - die `cons cells` erfassen Objekte mit fester Größe [c.f. `Lisp`] $\rightsquigarrow$ Sprachbausteine oder Parsing-Trees; jede solche Zelle umfasst $28\,\mathrm{Byte}$ auf einer $32\,\mathrm{bit}$– und $56\,\mathrm{Byte}$ auf einer $64\,\mathrm{bit}$–Architektur
- wenn ein neues Objekt erzeugt wird, sich ein existierendes nach einer Zuweisung in der Größe ändert, oder wenn ein neuer

Evaluation frame erzeugt wird, versucht R den Speicherbedarf aus diesem Arbeitsspeicher zu decken

- intern werden dazu die malloc–Funktionen verwendet
- werden bestimmte Schwellwerte in der Speicherauslastung überschritten, versucht R den Speicher aufzuräumen, um so wieder genügend freien Speicher zu haben; die Prozedur, die das erledigt heißt *Garbage Collection*
- die "Grundausstattung" an Speicher wird beim Aufruf von R vereinbart:
  - Kommandozeilenoption −−**min**−nsize:
    steuert die verfügbare Zahl an `cons cells`
  - Kommandozeilenoption −−**min**−vsize:
    steuert die verfügbare Größe des Heap in Bytes
  - diese "Grundausstattung" wird nie unterschritten
  - Konvention der Einheiten: $[N]$ $[<\text{Einheit}>]$
    * $[N]$ ohne Einheit $\hat{=}$ Zahl

  - $*$ $[N]$ k (übliches "kilo") $\hat{=}$ $[N] \times 1000$
  - $*$ $[N]$ K (Computer "kilo") $\hat{=}$ $[N] \times 2^{10} = [N] \times 1024$
  - $*$ $[N]$ M (Computer "Mega") $\hat{=}$ $[N] \times 2^{20} = [N] \times 1048576$
  - $*$ $[N]$ G (Computer "Giga") $\hat{=}$ $[N] \times 2^{30} = [N] \times 1073741824$
- in Windows: Kommandozeilenoption `--max-mem-size`:
  - Maximalausstattung — wird nie überschritten
  - per default auf dem Minimum aus Arbeitsspeicher (RAM) des Rechners und $256\,\mathrm{MB}$
  - minimal $10\,\mathrm{MB}$

- R–Befehle zum Speichermanagement während einer Session: siehe **help**(Memory)
  - **gc**: damit ruft der Benutzer die Garbage Collection explizit auf; falls Argument `verbose` auf `TRUE`, gibt **gc** anschließend einen Statusbericht über die Speicherverwendung ab
  - **gcinfo**: hiermit setzt man eine Flagge, ob man über eine ausgeführte Garbage Collection informiert werden will

- **gctorture**(**on** = TRUE): für Testzwecke (habe ich in meinem C–Interface alle Objekte auch wieder freigegeben?) — erzwingt eine Garbage Collection nach fast jeder Speicheranforderung; macht aber auch das System sehr langsam
- **memory.profile**: gibt die Speicherverwendung nach Typen [genauer SEXPREC] getrennt aus
- **memory**.size: gibt die maximale (falls Argument max auf TRUE) oder aktuelle (sonst) Speicherallokation durch malloc aus
- **memory**.limits: gibt die aktuellen Speicherbelegungsgrenzen an (Argument NA) bzw. versucht diese zu vergrößern (auf Argument size)

ACHTUNG Speicherbelegungsgrenzen können nur vergrößert werden!

- seit R Version 1.2.0 verwendet R einen "generationellen" Garbage Collector (**gc**)

- dieser verwaltet auch den Speicher dynamisch, d.h. er vergrößert und verkleinert den verfügbaren Speicher bei Bedarf (bis zu einem vorgegebenem Limit)
- Details siehe Tierney (2003b):
  http://www.stat.uiowa.edu/~luke/R/gengcnotes.html
- Idee: ein hierarchisches **gc** räumt zuerst bei den "jüngeren" Objekten auf
- es werden **keine** Speicherbereiche verschoben ("*non moving strategy*")
- daher: bei sukzessivem Füllen/Löschen des Speichers wird dieser immer stärker zerstückelt
- R–Beispiel 8.3-7 [Speicherprobleme]:
  Code verfügbar unter http://www.uni-bayreuth.de/departments/
  /math/org/mathe7/rkurs/speicher.r

```
library(boot)
```


```
data(nuclear)
nuke <- nuclear[,c(1,2,5,7,8,10,11)]
nuke.lm <- glm(log(cost)~date+log(cap)+ne+
           ct+log(cum.n)+pt, data=nuke)
nuke.diag <- glm.diag(nuke.lm)
nuke.res <- nuke.diag$res*nuke.diag$sd
nuke.res <- nuke.res-mean(nuke.res)
nuke.data <- data.frame(nuke,resid=nuke.res,
             fit=fitted(nuke.lm))
new.data <- data.frame(cost=1, date=73.00, cap=886,
            ne=0, ct=0, cum.n=11, pt=1)
new.fit <- predict(nuke.lm, new.data)
nuke.fun <- function(dat, inds, i.pred,
            fit.pred, x.pred)#
  {assign(".inds", inds, envir=.GlobalEnv)
   lm.b <- glm(fit+resid[.inds] ~date+
       log(cap)+ne+ct+log(cum.n)+pt, data=dat)
```


```
    pred.b <- predict(lm.b,x.pred)
    remove(".inds", envir=.GlobalEnv)
    c(coef(lm.b), pred.b-
      (fit.pred+dat$resid[i.pred]))
  }
for (i in 1:20)
  print(system.time(boot(nuke.data,
        nuke.fun, R=999, m=1,
        fit.pred=new.fit, x.pred=new.data)))
```

8.3.10 (b) Was passiert beim Einrichten eines neuen Evaluation Frame?

- Situation: ein Aufruf (Call) einer Funktion liegt vor und soll abgearbeitet werden
- Referenz: Chambers (1998, Abschnitt 4.9)

- dies geschieht in den folgenden vier/fünf Schritten
  1. New.**frame** erzeugt einen neuen Evaluation Frame für den Aufruf
     - als Element `N` einer `Frames`–Liste, in der die verschiedenen Frames verwaltet werden
     - dieser Evaluation Frame ist wieder ein `S`–Objekt, für das nun Speicher alloziert wird
     - parallel zur `Frames`–Liste "leben" in R noch eine `Calls`–Liste mit den entsprechenden Aufrufen und eine `Functions`–Liste mit den Funktionsdefinitionen
  2. der Auswertungsmechanismus ordnet die *tatsächlichen* Argumente des Aufrufs den *formalen* der Funktionsdefinition zu — vgl. Seiten 306ff.
  3. wenn Methoden für diese Funktion definiert sind — also die Funktion *generisch* ist, sucht der Auswertungsmechanismus die am besten passende Methode heraus
  4. **eval**(**body**,N) wertet den Körper (*body*) der Funktion /

Methode aus

- funktioniert analog zum Befehl
  eval(<expr>,<eval.Frame>)
- muss dabei womöglich wieder bei Zuweisungen und weiteren Aufrufen neuen Speicher allozieren

5. Clear.**frame** wird aufgerufen
   - Zweck: den Frame N wieder zu löschen
   - davor wird aber der Wert des Körpers — als Rückgabewert der Funktion / Methode in den Vater–Frame Parents[N] befördert

- lazy evaluation — vergleiche auch Seite 307 — bewirkt, dass Speicher für lokale Kopien ("Call by Value"!) nur dann angefordert wird, wenn das Argument im Körper zur Auswertung eines Ausdrucks / Aufrufs gebraucht wird

8.3.10 (c) Wann muss ein Objekt ausgewertet werden?

- Referenz: Chambers (1998, Abschnitt 4.9)
- die Möglichkeit, unausgewertete Ausdrücke ("promises", c.f. Seite 307) zu übergeben, erspart beim "Call by Value" das Anlegen vieler Kopien
- genauer muss `S` eigentlich nur in folgenden drei Situationen Ausdrücke auswerten
  1. bei der Übergabe von Objekten an `C`, `C++`, `Fortran`,....; insbesondere benötigen (in `C`/`Fortran` vercodete) elementare arithmetische, logische Operationen den tatsächlichen Wert des Objekts
  2. beim Method Dispatching — vgl. Abschnitt 8.1.3 (c); hier wird aber nur der Typ benötigt; Missings sind hier zugelassen — i.a. nicht in 1.
  3. bei der Zuweisung, wenn es tatsächlich "kopiert" wird — s.u.

- S — und damit auch R haben das Prinzip des *Data Sharing*
  - nach Obengesagtem wird der Wert einer (lokalen Kopie) einers Objekts in einem Evaluation Frame nur durch Zuweisungen verändert;
  - solche Zuweisungen umfassen insbesondere auch *Ersetzungsmethoden* — vgl. Abschnitt 8.1.5 (e); ebenso mehr oder weniger analog: assign– oder **eval**–Ausdrücke
  - um beim ("Call by Value"!) nicht zu viele (unnötige) lokale Kopien anlegen zu müssen, verwendet R das Prinzip des *Data Sharing*; diese funktioniert so:
    * jedes Objekt in S hat einen internen Referenzzähler;
    * dieser zählt, wie oft dieses Objekt innerhalb eines Frames rechts von einer Zuweisung gestanden ist — explizit oder als Element / Slot eines Objekts.
    * bei Übergabe eines Arguments in einem Funktionsaufruf wird bei Auswertung dieses Arguments in der Funktion dieses im neuen Evaluation Frame zugewiesen

    - dabei werden die Referenzzähler des Objekts und aller seiner Elemente / Slots um 1 erhöht, aber nicht wirklich eine Kopie angelegt
    - erst wenn etwas am Objekt geändert wird — z.B. mit einer Ersetzungsmethode wird wirklich eine Kopie angelegt, — bei komplexeren Strukturen nur eine Kopie des entsprechenden Elements / Slots
    - beim Verlassen des Frames wird dann der Evaluation Frame gelöscht — mit den evtl. angelegten tatsächlichen Kopien, der entsprechende Speicher freigegeben und der Referenzzähler um eins [oder mehr] verringert
- diese Technik funktioniert gut bei Iterationen und nur sehr schlecht bei Rekursionen;
    - wenn daher ein rekursiver Algorithmus auch gut als Iteration zu formulieren ist, sollten Iterationen verwendet werden
    - bei Algorithmen wie `quicksort` ist dies nicht so gut

möglich $\rightsquigarrow$ Trade–Off zwischen guter Lesbarkeit der Rekursion und Speicherersparnis bei Iteration

  * — bei `quicksort` $\mathrm{O}(n \log(n))$ zu $\mathrm{O}(n)$, also in den meisten Fällen Rekursion vertretbar

- *Data Sharing* auch relevant bei Schnittstellen zu `C` / `Fortran`
  * per Default geht `S` auf Nummer sicher und legt Kopien an
  * um dies zu vermeiden: man kann der Schnittstelle mitteilen, dass ein `S`-Objekt in `C` / `Fortran` nicht verändert werden wird $\rightsquigarrow$ `COPY`–Argument der Schnittstelle

8.3.11 Verschiedene Aufrufe von R und Kommandozeilenoptionen

8.3.11 (a) Ausführmodi von R

- interaktiver / Sitzungs–Modus: R wartet auf Eingaben an der Console
- BATCH–Modus: R verarbeitet ein Script und beendet sich anschließend, ohne auf Eingaben zu warten
- unter Windows wird der Modus durch Verwendung eines der beiden `.exe`–Files `RGui.exe` und `Rterm.exe` entschieden — siehe nächster Abschnitt
- unter Linux wird generell der Sitzungs–Modus verwendet, es sei denn man ruft `R CMD BATCH` auf — siehe nächster Abschnitt

- generell: — vgl. An Introduction to R (2006a, Anhang B)
  - beim Aufruf von R können verschiedene Optionen bereits über die Kommandozeile gesetzt werden
  - der konkrete Aufruf unterscheidet sich je nach Betriebssystem
  - Optionen die bei allen Betriebssystemen funktionieren
    - `-version`: gibt die Versionsnummer aus und beendet die Session
    - `-save`/`-no-save`: kontrolliert, ob beim Beenden der Session der Arbeitsspeicher abgespeichert werden soll; wenn keine dieser Optionen angegeben wird, wird im interaktiven Modus beim Beenden der Sitzung danach gefragt; im BATCH–Modus muss eine der beiden angegeben werden
    - `-no-site-file`: kein Laden eines `site`-weiten Initialisierungsfiles

* `-no-init-file`: kein Laden eines benutzerdefinierten Initialisierungsfiles
* `-no-environ`: unterdrückt das Laden von Umgebungsvariablen
* `-restore`, `-no-restore`, `-no-restore-history`, `-no-restore-data`: kontrolliert, ob der Zustand der letzten Sitzung (oder zumindest die Daten) wiederhergestellt werden sollen
* `-vanilla`: Kombination aus `-no-save`, `-no-restore`, `-no-site-file`, `-no-init-file` und `-no-environ`
* `-min-vsize=N`, `-min-nsize=N`, `-max-vsize=N`, `-max-nsize=N`: siehe Abschnitt 8.3.10 (a)
* `-quiet`, `-silent`, `-q`: keine Startup Meldung
* `-slave`: R so still wie möglich laufen lassen
* `-verbose`: Ausgabe von detaillierter Ablauf–Information; insbesondere wird in **options** `verbose` auf `TRUE` gesetzt

- Zugriff auf die Kommandozeilenparameter in R mit **commandArgs**
- Ablauf des "Hochfahrens" von R
  * Bei Aufruf ohne `-no-environ` wird aus Umgebungsvariable `R_ENVIRON` der Pfad auf File mit `site`–weiten Umgebungsvariablen entnommen; ist `R_ENVIRON` undefiniert, verwendet R `$R_HOME/etc/Renviron.site`, so dieses existiert
  * die benutzersdef. Einstellungen werden aus File `.Renviron` im Stammverzeichnis des jew. Benutzers geladen
  * diese beiden Files enthalten Zeilen der Form `<name>=<wert>`; siehe auch **help**(Startup); wichtige Variablen dabei:
    · `R_PAPERSIZE` für die Default Papiergröße zum Drucken
    · `R_PRINTCMD` für den Default Druckbefehl
    · `R_LIBS` mit den Verzeichnissen, die nach R–Zusatzpaketen durchsucht werden sollen

  - R_PROFILE der Name des site–weiten startup–Profil

* dann sucht R das site–weite startup–Profil im File aus der Variablen R_PROFILE, es sei denn Option -no-site-file ist angegeben; falls keine Variable R_PROFILE existiert, verwendet R $R_HOME/etc/Rprofile.site, sofern dieses existiert
* dann sucht R das benutzereigene startup–Profil im File .Rprofile im Stammverzeichnis des aktuellen Benutzers und lädt es mit **source** ein, es sei denn Option -no-init-file ist angegeben;
* außerdem wird das File .RData im aktuellen Verzeichnis eingeladen, es sei denn Option -no-restore oder -no-restore-data ist angegeben;
* schließlich wird die Funktion .First aufgerufen, sofern diese existiert; diese und die Funktion .Last, die am Abschluss einer Sitzung ausgeführt wird, kann man in den geeigneten Startup–Files oder im File .RData ablegen

- daneben auch Aufruf von speziellen R–utilities mit **R** CMD <command> <args> in Linux, bzw. Rcmd <command> <args>; an Kommandos gibt es
  - BATCH: lässt R im BATCH Modus laufen
  - INSTALL: installiert Zusatzpakete
  - REMOVE: löscht Zusatzpakete
  - build: "baut"/legt Zusatzpakete an— vgl. Abschnitt 8.2.9
  - check: überprüft Zusatzpakete — vgl. Abschnitt 8.2.11
  - Rprof: profiliert ein R–File, siehe Abschnitt 8.3.2
  - Rdconv: konvertiert Files im .Rd–Format in verschiedene andere Formate wie HTML, Nroff, LaTeX, puren ASCII–Text und S–Dokumentations–Format
  - Rd2dvi.sh: konvertiert Files im .Rd–Format in .dvi/.pdf–Format
  - Rd2txt: konvertiert Files im .Rd–Format in puren puren ASCII–Text
  - Sd2Rd: konvertiert Files im S–Dokumentations–Format in

.Rd–Format

  - Hilfe ist mit **R** CMD <command> -help bzw. Rcmd <command> -help verfügbar

- Unterschiede Windows/Unix
  - Syntax in Windows
    * es gibt zwei verschiedene Aufrufe:
      · RGui.exe: "graphische Benutzeroberfläche" — ruft man normalerweise auf
      · Rterm.exe: terminal–Version — für BATCH–Aufrufe
    * beide können mit Optionen den oben genannten gestartet werden
    * bei Rterm.exe möglich: zusätzliches Setzen von Pipes für Ein– und Ausgabefile, also Rterm[options][< infile][>outfile]
    * spezielle Windows–Optionen:
      · -mdi, -sdi, -no-mdi: kontrolliert, ob mehrere Fenster als Unterfenster eines Hauptfenster (mdi) oder nur

Hauptfenster (`sdi`) für die Konsole, Graphic, Reports geöffnet werden sollen
  - `-max-memsize=N`: siehe Abschnitt 8.3.10 (a)
  - `-ess`: Konfiguriere R zur Zusammenarbeit mit `EMACS` im `R-inferior-mode`
* spezielle Umgebungsvariablen für R in Windows regeln das R–Stammverzeichnis:
  - mit höchster Priorität `R_USER` — falls definiert
  - dann `HOME` — falls definiert
  - sonst `HOMEDRIVE+HOMEPATH` — falls definiert (normalerweise bei `Win NT/2K` der Fall)
  - sonst ist das Stammverzeichnis (üblicherweise `C:\`) R–Stammverzeichnis:

– Syntax in `Linux`
* **`R[options][< infile][>outfile]`**
* spezielle Linux–Optionen:
  - `-help`, `-h`: gibt eine kurze Hilfe als Nachricht aus und

beendet die Session

- RHOME: gibt das R–Stammverzeichnis aus und beendet die Session
- -no-readline: schaltet das Kommandozeilen–Editieren über readline aus; nützlich für Nutzung von Emacs und ESS
- -debugger=<name>, -d <name>: lässt R unter einem Debugger <name> ablaufen
- -gui=type, -g type: benutzt type als graphische Benutzeroberfläche; aktuell vorgesehen sind X11, gnome und none

* spezielle Kommandos für **R** CMD in Linux:
  - COMPILE: compiliert Files für R — z.B. C–Files mit Code für .**C**
  - SHLIB: legt Shared Library an — vgl. Abschnitt 8.3.8
  - LINK: linkt Files für R — Front–End für ausführbare Programme

8.3.12 R auf Parallelrechnern

8.3.12 (a) Nutzung von für Multiprozessorbetrieb optimierter Infrastruktur

- numerische Infrastruktur von R:
  öffentliche Bibliotheken aus der Netlib
- von dort: insbesondere Routinen zur linearen Algebra (BLAS —Basic_Linear_Algebra_Subprograms)
- R<1.7.0: Bibliotheken EISPACK und LINPACK
- aber: diese basieren (nur) auf Level 1 BLAS, d.h. nicht effizient auf Multi–Prozessor Rechnern mit gemeinsamen Speicher

$\Longrightarrow$ R≥1.7.0 Umstieg auf LApack (BLAS level 3 konform)

- auch möglich: Nutzung von system-optimiertem BLAS:
  ATLAS — Automatically Tuned Linear Algebra Software

$\Longrightarrow$ Welche (am besten vorcompilierte) Version von ATLAS für welche Architektur? Hinweise dazu:

http://www.kevinsheppard.com/research/matlabatlas/matlab_atlas.aspx

- unter Linux: Steuerung beim Compilieren über Option
  --with-blas="<BLAS-Linkspezifikationen>",
  vgl. R Installation and Administration (2006d, Abschnitt A.2.2)
- für Windows: Archiv für viele vorcompilierte ATLAS-DLL's für verschiedene Architekturen auf CRAN unter
  bin/windows/contrib/ATLAS
- eigene Optimierung:
  1. Klärung: Welche Version von ATLAS ist die richtige?
  2. Herunterladen dieser Version
  3. Überschreiben der bestehenden Datei Rblas.dll im Verzeichnis R_HOME_bin

8.3.12 (b) "Poor Man's Parallel" — händische Verteilung auf Knoten

- aus heutiger Sicht veraltet; aber didaktisch interessant;
- entstanden im Rahmen der Diplomarbeit von *Florian Camphausen*

- Vorbereitungen an R-Code ...
- Übergabe von Parametern an R-Prozess
  - Aufruf (in `csh`) mit

    R **$**i −−restore −−save −−no−readline −−gui=none < in.R > out**$**i;

    * ruft R mit Übergabeparameter "Inhalt der Variable `i`" auf
    * weitere Optionen wie in Abschnitt 8.3.11 (b) beschrieben
    * das File `in.R` enthält den abzuarbeitenden Code
    * das File `out$i` (wobei `$i` durch den Inhalt der Variable `i` ersetzt wird) wird mit allen Ausgaben von `in.R` gefüllt
  - Zugriff auf die Kommandozeilen-Übergabeparameter in R mit

    as.integer(commandArgs()[2])
- Steuerung der einzelnen R-Prozesse durch drei geschachtelte shell-scripten; Hintergrund:

- aus "politischen" Gründen sollen nicht mehr als n Knoten simultan angefordert werden

$\rightsquigarrow$ Teile Problem der Ordnung M auf in $m = M/n$ Scheiben/Blöcke

- fordere also m mal n Knoten an
- was innerhalb Scheibe `s` von Prozessor j zu tun ist, wird im R-Code `in.R` anhand des beim Aufruf übergebenen Parameters `$i`$= (\texttt{s} - 1) \times n + j - 1$ entschieden
- kleiner Punkt zur `csh`-Syntax: Rückgabe des Werts der Variable `<Varname>` mit **`$`**`<Varname>`, numerische Berechnungen mit `@`

$\rightsquigarrow$ shell-script `Alltask` mit einer Schleife, die die Scheiben 1 bis m durchläuft und jeweils einen Block von n Knoten parallel anfordert (mit `qsub`) und beschickt

- Quelle unter `AllTask`

```
#!/bin/tcsh
#
date; qstat
@ I = 1
while ( $I <= $1)
 echo $I te Prozess-Scheibe
 qsub -v sc=$I,wieviel=$2 -l nodes=$2 Task
 @ I = $I + 1
end
qstat; date
```

- Aufruf: ./AllTask <nS> <nK> > <Protokoll>
- `<nS>` (#Scheiben) wird mit `$1`, `<nK>` (#Knoten) mit `$2` an `qsub` mit Option `-v` übergeben
- `qsub` regelt die Mitprotokollierung sämtlicher Ausgaben von `Task` im Kontrollfile `Task.o<prozessname>`
- Ausgabe von `Alltask` wird in File `<Protokoll>` geschrieben

⇝ shell-script `Task`, das innerhalb einer Scheibe insgesamt n R-Prozesse unter Verwendung von PBS beschickt

– Quelle unter `Task`

```
#!/bin/tcsh
#PBS -l walltime=24:00:00
#PBS -j oe
#
cd <hier steht ein lokaler Pfad> ; date
set n=`cat $PBS_NODEFILE`
@ i = ( $sc - 1 ) * $wieviel + 1
@ j = 1
while ( $i <= $sc * $wieviel )
 echo $i
 echo "Task $j in Sch. $sc auf Rechner $n[$j]"
 rsh $n[$j] < myRcall &
 @ i = $i + 1
 @ j = $j + 1
end
sleep 60; date
```


```
#
while ( `ls -1 *.$sc.run|wc -l` )
    sleep 60
end
mail <....> ; date
# und Tschuess....
exit
```

- # Prozessoren bereits von `qsub` in `AllTask` gesetzt
- **cat** $PBS_NODEFILE schreibt in Variable `n` den Vektor der zugeteilten Knoten
- `rsh` öffnet Shell auf Knoten **$**`n`[**$**`j`] (Indizierung von `n` wie in `C`), auf der das Eingabefile `myRcall` ausgeführt wird;
- in `myRcall`:
  vor Aufruf von R: Anlage d. Kontrollfiles `Task.o<prozname>`
  nach Rückkehr von R: Löschung
- `&` schiebt den `rsh` in den Hintergrund; es wird also nicht bis zu dessen Terminierung gewartet

- mit date: Protokollierung von Uhrzeit und Datum
- erstes sleep zur Sicherheit, falls Fileserver zu langsam
- Sinn der zweiten **while** Schleife:
  * anhand der Kontrollfiles wird geprüft, wie lange der Prozess `Task` noch "leben" muss, bis man ihn mit **exit** verlassen darf:
  * so lange, bis alle Kontrollfiles der Scheibe gelöscht sind
  * ls -1 *.**$**sc.run listet alle Kontrollfiles dieser Scheibe, welche anschließend mit wc ("Wordcount") gezählt werden
  * bis wc 0 Zeilen zählt, wartet man immer wieder weitere 60 Sekunden

⇝ Textdatei `myRcall` mit eigentlichem R-Aufruf

- Quelle unter `myRcall`

```
cd `pwd`
echo "'$$'" > $n[$j].$sc.run
R $i --restore --save --no-readline \
     --gui=none < alles.R > erg$i
rm $n[$j].$sc.run"
```

- der Aufruf von **echo** legt das Kontrollfile `$n[$j].$sc.run` an, z.B. `node29.3.run`, und schreibt Prozessnummer hinein

• Diagnostik während des Ablaufs

- für Einsicht in die Warteschlange (Queue), die `qsub`: `qstat -a`; vgl. auch `man qstat`
- für Löschungen von Einträgen in der Warteschlange: `qdel`; vgl. `man qdel`

8.3.12 (c) fertige Pakete zum parallelen Rechnen

[basiert auf Referat "Paralleles Rechnen mit R"von *Lukas Gudmundsson* am 14.09.2006]

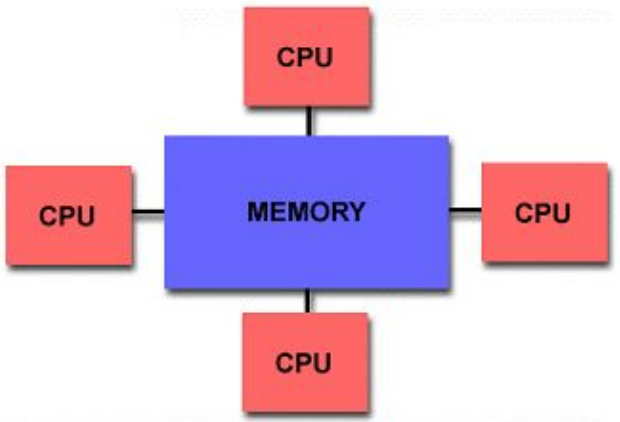
- Motivation: einige Beispiele von Anwendungen rechenintensiver, statistischer Verfahren

Problem	Verfahren	Anwendugen
Mustererkennung / maschinelles Lernen	Neuronale Netze / Support Vector Machines (SVMs) / Data-Mining	´ Analyse von Kundendaten, Genetik: Microarray-Analysen, Pharmabereich: Wechselwirkungen von Medikamenten
Untersuchung komplexer Wechselwirkungen / Dimensionsreduktion	stochastische / numerische Simulation, Gibbs-Sampler (MCMC)	´ Meteorologie, Klimatologie, Geologie, Fahrzeugbau (Crashtests), Genetik: Sequenzierung,

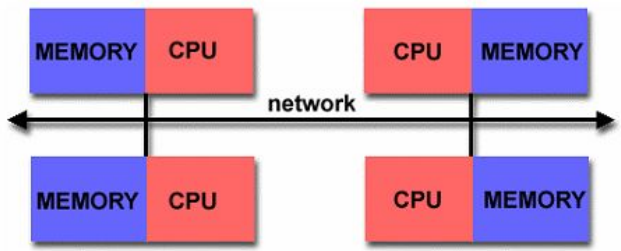
Problem	Verfahren	Anwendugen
robuste Kovarianzen hochdimensionaler Daten	Resampling-Techniken (Bootstrap)	´ Ausreißeridentifikation, Sensitivitätsanalysen statistischer Verfahren

- Situation: eine Aufgabe ist "automatisiert" bestmöglich auf mehrere CPUs zu verteilen
- Problemvoraussetzung: *Parallelisierbarkeit*
  - mehrere Teilaufgaben können (weitgehend) unabhängig von einander parallel gelöst werden
  - dazu nötig: geringer Kommunikationsbedarf der Teilaufgaben untereinander

- Hardware-Architekturen
  - Multiprozessor-Maschine
    - ein Speicher
    - Datenaustausch über Speicher
    - (eine R - Version)



  - Cluster von Maschinen
    - verschiedene Architektur
    - Datenaustausch über Netzwerk
    - (verschiedene R - Versionen)



- Software zur Regelung der Kommunikation und Verwaltung der Knoten
  - `MPI` (Message Passing Interface)
    * Quasi-Standard
    * fortlaufend in Weiterentwicklung
    * hervorragend für Multiprozessor-Maschinen
  - `PVM` (Parallel Virtual Machine)
    * weit verbreitet
    * konzeptionell für heterogene Cluster von Maschinen
    * seit ca. 1997 nicht weiterentwickelt
  - Verfügbarkeit/Installation
    * beide Konzepte als Pakete im Open Source Projekt implementiert verfügbar
    * Installation bei Multiprozessor-Maschinen nur einmal nötig, bei heterogenen Netzen u.U. mehrfach (nicht bei homogenen Netzen mit gemeinsamen Fileserver)

- R-Pakete zur Parallelisierung
  - `RPVM` (Na Li and Rossini (2001); liefert Infra-Struktur)
    - Vorbereitung: Installation von `PVM`, `Setzen von Pfaden`
    - Shell-Skripten zur Kommunikation zwischen `PVM` und R
    - Starten der Prozesse in R
    - Low-level und high-level Tools zur Parallelisierung
  - `Rmpi` (Yu (2002); liefert Infra-Struktur)
    - Vorbereitung: Erstellen eines `Beowulf` Clusters; Installation von `LAM-MPI`
    - Starten von R-Slaves aus R heraus
    - `Rmpi` implementiert mehrere `MPI`-funktionen zu Parallelisierung in R
  - `snow` (Rossini et al. (2003); komfortables User-Interface)
    - Basiert auf rpvm oder Rmpi
    - Vor Beginn einer Sitzung: Initialisierung von `PVM` / `MPI`

* Wichtige Funktionen in `snow`
  - *Administrative Routinen*
  - makeCluster: erzeugt einen neuen Cluster
  - stopCluster: schließt Cluster
  - clusterSetupSPRNG: Initialisierung für parallele Zufallszahlen
  - *High Level Routinen*
  - parLapply: paralleles **lapply**
  - parSapply: paralleles **sapply**
  - parApply: paralleles **apply**
  - *Basic Routines*
  - clusterExport: exportiert Variablen zu Knoten
  - clusterCall: ruft Funktion in jedem Knoten auf
  - clusterApply: wendet Fkt. auf Argumente in jedem Knoten an
  - clusterApplyLB: clusterApply mit ausgeglichener Aufgebenverteilung
  - clusterEvalQ: Auswertung eines Ausdrucks auf Knoten
  - clusterSplit: Aufteilung eines Vektors an Knoten

R–Beispiel 8.3-8 [Beispiel mit snow und Rmpi]:

```
require(boot); require(snow}

# erzeugt ein Cluster aus 4 Maschienen
cl ← makeCluster(4, type = "MPI")

# wertet library(boot) auf allen Maschinen aus
clusterEvalQ(cl, library(boot))

# definition der zu bearbeitenden Funktion
ratio ← function(d, w) sum(d$x * w)/sum(d$u * w)

# ausführen des Bootstrap auf dem Cluster
clusterCall(cl,boot,city,ratio, R=999,stype="w")

# Stopt den Cluster
stopCluster(cl)
```


- Quellen
  - allgemeine Einführungen in paralelles Rechnen:
    http://www.llnl.gov/computing/tutorials/parallel_comp/,
    http://www.numa.uni-linz.ac.at/Staff/haase/parvor_e/node1.html
  - Links zu MPI: http://www.mpi-forum.org,
    http://www.lam-mpi.org
  - Links zu PVM: http://www.netlib.org/pvm3/,
    http://www.csm.ornl.gov/pvm/pvm_home.html
  - MPI und PVM im Vergleich: Geist et al. (1996)
  - Unix - Multiprozessoren Netzwerk Beowulf:
    http://www.beowulf.org
  - Einführung zum Parallelen Rechnen in R:
    Abstract und Folien zu Vortrag von
    Justin Harrington und Matias Salibian-Barrera

- `snow` von Luke Tierney, A.J. Rossini, Michael Na Li, H. Sevcikova, http://cran.us.r-project.org/src/contrib/Descriptions/snow.html;
  weitere Informationen unter Rossini et al. (2003) und
  http://www.stat.uiowa.edu/~luke/R/cluster/cluster.html,
  sowie unter http://www.sfu.ca/~sblay/R/snow.html
- `pvclust` von Ryota Suzuki und Hidetoshi Shimodaira,
  http://cran.us.r-project.org/src/contrib/Descriptions/pvclust.html;
  weitere Informationen unter
  http://www.is.titech.ac.jp/~shimo/prog/pvclust/
- `Rmpi` von Hao Yu, http://cran.us.r-project.org/src/contrib/Descriptions/Rmpi.html;
  weitere Informationen unter Yu (2002) und
  http://www.stats.uwo.ca/faculty/yu/Rmpi
- `rpvm` von Na Li und A.J. Rossini,
  http://cran.us.r-project.org/src/contrib/Descriptions/Rpvm.html;
  weitere Informationen unter Na Li and Rossini (2001)

8.3.13 Beispiel: R im InterNet — R im BATCH–Modus

Beispiel von *Matthias Kohl*:

Routine zur Berechnung des Minimax–Radiuses

. . . z.Z. nicht online ausführbar, weil Rechner vom Netz, auf dem Web-Server und R liefen. . .

- Bereitstellung eines HTML–Formulars als Benutzerschnittstelle
  - c.f. http://www.uni-bayreuth.de/departments/math/org/mathe7/radius/program.html
  - Benutzer: radius; Kennwort: unknown

R–Beispiel 8.3-9 [der HTML-Code]:

```
<!DOCTYPE html
  PUBLIC "-//w3c//dtd␣html␣4.0␣transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
```


```
␣␣␣␣charset=iso-8859-1">
  <meta name="Author" content="Matthias␣Kohl">
  <title>The Costs of Not Knowing the Radius</title>
</head>
<body background="../papier.gif">
Previous page:
<a href="../radius/index.html">
   Mathematical Statistics (MATHE  VII) - radius
</a>

<hr><img src="../logo.gif" alt="[Picture:␣logo]">
<br><a name="top"></a>

<div align="Center"><h1><font size="+3">
   The Costs of Not Knowing the Radius
</font></h1></div>

<hr><br>
<div align="Center"><b><big>
```


```
    Program for k-dimensional location
</big></b><br></div>
<br><br>
<center><b>
   This program determines the least favorable radius
   r<sub>0</sub> for a given radius interval
   [r<sub>a</sub>, r<sub>b</sub>].
</b></center>
<br><br>
<!-- ================================================== -->
<!-- Angabe Dimension + Intervall [start, ende]         -->
<!-- ================================================== -->
<form action="$PFAD/prgm.cgi" method="post">
<pre>
  <b><font size="+1">
     Choose a dimension k between 1 and 25:
  </font></b>
  <input name="k" size="2" maxlength="2"> </input>
</pre>
```


```
<br>
<b> <font size="+1">
    Specifiy a radius interval [r<sub>a</sub>,
    r<sub>b</sub>],   with r<sub>a</sub> in [0, 5)
    and r<sub>b</sub> in (0, 5].
<br>
    For unbounded radius intervals set r<sub>b</sub>
    to 6!
</font></b>
<br>
<pre>
  <b><font size="+1">
     Left limit r<sub>a</sub> of the radius interval:
  </font></b>
  <input name="start" size="4" maxlength="4"> </input>
</pre>
<br>
<pre>
  <b><font size="+1">
```


```
      Right limit r<sub>b</sub> of the radius interval:
  </font></b>
  <input name="ende" size="4" maxlength="4"> </input>
</pre>
<br>
<br>
<input type="submit" value="Submit">
<input type="reset"  value="Cancel">
</form>
<!-- ================================================== -->
<br>
<hr> This page is maintained by
 <a href="mailto:Matthias.Kohl@uni-bayreuth.de">
      Matthias Kohl</a>
<br> Last modified: $Date: 2001/10/10 $
</body>
</html>
```


- Auswertung der Eingabe in das Formular

  Übergabe per HTML–Methode "post" über stdin an cgi–Skript, einem ausführbaren C–Programm, das den Eingabestrom filtert (eigentlich besser geeignet: Perl), R aufruft und schließlich dynamisch Antwort-HTML-Seite erzeugt

  R–Beispiel 8.3-10 [der C-Code]:

  verfügbar unter http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/prgm.c

```
# include <math.h>
# include <stdio.h>
# include <stdlib.h>
# define buffer 16

void filecopy(FILE *ifp,FILE *ofp);
/* kopiert File *ifp auf File *ofp */
void filter (char *** varlist, char *** inhaltlist,
             int *n);
```


```
/* filtert einen Tokenstrom auf stdin
   in eine Liste vom Typ Eintragname=Eintrag */

int main (void)
 {int i=0,n=-1;
  char ** vl, **il;

/* dynamische Erzeugung einer HTML Seite
   auf stdout */
  printf("Content-type: text/html%c%c",10,10);
  printf("<!DOCTYPE HTML PUBLIC>\n");
  printf("<HEAD>\n");

/* ..<weggelassen>... */

  printf("</HEAD>\n");
  printf("<body>\n");
```


```
/* ..Hintergrund und Kopf der HTML-Seite.. */
/* ..<weggelassen>... */

/* Filtern des Tokenstroms auf stdin */

  filter(&vl, &il, &n);

    /* dynamisches Schreiben eines R-Skripts
       hier PFAD-Variablen aus Sicherheitsgruenden
       nicht offengelegt
     */

     fo=fopen("$PFAD/make.r","w+");

  if(n>=0)
   {
/* Ueberpruefung der Eingaben von stdin */
```


```
  int dim=atoi(il[0]), test;
  double start=atof(il[1]), ende=atof(il[2]);
  int len0, len1, len2;
  char *aufruf; /*Unix--Kommando*/
  char c;
  FILE *fo;

  len0=strlen(il[0]);
  len1=strlen(il[1]);
  len2=strlen(il[2]);

/* Eingabefehler abfangen ..<weggelassen>... */

/* als erstes wird im dynamisch erzeugten
   R-Skript das eigentliche R-Source File
   per source eingelesen    */

  fprintf(fo,"␣%s\n␣",
        "source(file=\"$PFAD/radius.r\")");
```


```
/* Aufruf der eigenen Funktion 'optimal'
   mit den Eingabeparametern vom WWW-Server
*/

fprintf(fo,"%s", "optimal(k=");
fprintf(fo,"%d", dim);
fprintf(fo,"%s", ", start=");
fprintf(fo,"%f", start);
fprintf(fo,"%s", ", ende=");
fprintf(fo,"%f", ende);
fprintf(fo,"%s", ")\n");
fclose(fo);

/* aufruf ist String vom Typ

   aufruf="R BATCH $PFAD/make.r $PFAD/ausgabe.txt"

   hier PFAD-Variablen aus Sicherheitsgruenden
   nicht offengelegt        */
```


```
/* Aufruf von R im BATCH Modus mit R-Skript,
   Konsolenausgabe auf Datei ausgabe.txt
   dabei schreibt radius.r die Ergebnisse auf Files
   radius.txt, ineff.txt, dauer.txt
*/
 test=system(aufruf);

 /* Auslesen */
 fo=fopen("$PFAD/radius.txt","r");
 if(fo==NULL)
   printf("Datei konnte nicht goeffnet werden!");

 /* simultan: dynamischer Aufbau der Ergebnisseite
    mit den Resultaten durch Schreiben auf stdout
 */

 printf("<div align=\"center\">");
 printf("<h2>Program for k-dimensional location");
 printf("</h2><br>");
```


```
printf("<font size=\"+1\">");
printf("The results for dimension k=%d", dim);
printf("and radius interval");
printf("</font>");

/* Ausgabe Minimax Radius */

if(ende==6)
{
  printf("<font size=\"+1\">");
  printf(" [%1.2f, infinity]</font>", start);
}
else
{
  printf("<font size=\"+1\">");
  printf(" [%1.2f, %1.2f] </font>", start, ende);
}
```


```
printf("<font size=\"+1\">are:</font><br><br>");
printf("<font size=\"+1\">least favorable radius:");
printf("</font>");
while((c=getc(fo))!=EOF && c!='\n')
  {printf("<font size=\"+1\">");
   printf("<font color=\"#FF0000\">");
   printf(" %c </font></font>", c);}
fclose(fo);


/* Ausgabe Effizienzverlust */

fo=fopen("$PFAD/ineff.txt","r");
if(fo==NULL)
  printf("Datei konnte nicht goeffnet werden!");
```


```
printf("<br><br> <font size=\"+1\">");
printf("minimax inefficiency: </font>");

while((c=getc(fo))!=EOF && c!='\n')
  {printf("<font size=\"+1\">");
   printf("<font color=\"#FF0000\">");
   printf(" %c </font></font>", c);}
printf("</div>");
fclose(fo);

/* Ausgabe Rechenzeit */

fo=fopen("$PFAD/dauer.txt","r");
if(fo==NULL)
  printf("Datei konnte nicht goeffnet werden!");

printf("<br><br><br> computation time (sec):");
printf(" </font>");
```


```
while((c=getc(fo))!=EOF && c!='\n')
  {printf("<font size=\"+1\">");
   printf("<font color=\"#0000FF\">");
   printf(" %c </font></font>", c);}
printf("</div>");
fclose(fo);

/* Fuss und Maintainervermerk */

printf("<br><hr><br><div align=\"Left\">");
printf(" This page is maintained by ");
printf("<a href=\"mailto:");
printf("Matthias.Kohl@uni-bayreuth.de");
printf("Matthias Kohl</a><br>");
printf(" Last modified: $Date: 2001/10/10 $");
printf("\n </body>\n");

/* Speicherfreigabe */
```


```
      if(n>0)
       {
         for (i=0; i<=((n/buffer)+1)*buffer;i++)
           {
             free (vl[i]); free(il[i]);
           }
           free (vl); free(il);
       }
    }
    else
    {
       printf("\n␣</body>\n");
    }
  return(0);
}
```


- das eigentliche R-Programm — verfügbar unter

  http://www.uni-bayreuth.de/departments/math/org/mathe7/

  /rkurs/radius.r

```
##################################################
# Funktionen zur Berechnung des least fav.
# Radius im Fall k-dim. Lokation fuer ein
# vorgeg. Radiusintervall [start, ende]
##################################################

#Hilfsfunktion "gestutzte" Verteilungsfunktion
Ffkt <- function(k, t)
{
    erg <- pgamma((t^2/2), (k/2))
    return(erg)
}

#Hilfsfunktion gestutzter Erwartungswert
```


```
Efkt <- function(k, t)
{
    erg <- sqrt(2)*gamma((k+1)/2)/gamma(k/2)*
           pgamma((t^2/2), (k+1)/2)
    return(erg)
}

#Hilfsfunktion gestutzte Varianz
Vfkt <- function(k, t)
{
    erg <- k*pgamma((t^2/2), (k+2)/2)
    return(erg)
}

#Funktion zur Berechnung der optimalen
#IC als Loesung des MSE-Problems (p.207) im
#Lokationsmodell (P_0 = N_k(0,I))
```


```
#(vgl. Rieder (1994, Theorem 5.5.7(b)))
#unter Verwendung eines zweidimensionalen
#Newton-Verfahrens
lCoptk <- function(k, r, Astart=1, bstart=1,
                   delta=1e-9)
{
    A1 <- Astart
    b1 <- bstart
    A0 <- 0
    b0 <- 0

    #Ab-Schritt
    while(max(abs(A1-A0), abs(b1-b0)) > delta)
    {
        A0 <- A1
        b0 <- b1
        c0 <- b0/A0
```


```
g ← sqrt(2)*gamma((k+1)/2)/gamma(k/2)
f ← dgamma(c0^2/2, k/2)
e ← g*dgamma(c0^2/2, (k+1)/2)
v ← k*dgamma(c0^2/2, (k+2)/2)
Ff ← Ffkt(k, c0)
E ← Efkt(k, c0)
V ← Vfkt(k, c0)
G2 ← g - E - c0 + c0*Ff - r^2*c0
G2b ← 1/A0*(-c0*e - 1 + Ff +
        c0^2*f - r^2)
G2A ← 1/A0*(c0^2*e + c0 - c0*Ff -
        c0^3*f + r^2*c0)

detJ ← V*G2b - (g - E)*G2A
A1 ← (k*G2b + G2*(c0*v + g -
        E - c0^2*e))/detJ
```


```
        b1 <- -(k*G2A + G2*(V - c0^2*v +
                  c0^3*e))/detJ
    }
    erg <- c(A1, b1)
    return(erg)
}
#Hilfsfunktion zur Bestimmung des
#optimalen r0 innerhalb eines bestimmten
#Bereichs [begin, end]
intervall <- function(r, k, rli, rre, delta)
{
    Ab <- ICoptk(k, r, 1, 1, delta)

    if(rre == 6)
    {
        bmin <- k/sqrt(2)/gamma((k+1)/2)*gamma(k/2)
    }
```


```
    effre <- Ab[2]^2/bmin^2
  else
  {
    Abre <- ICoptk(k, rre, 1, 1, delta)
    effre <- (k*Ab[1] - Ab[2]^2*(r^2 - rre^2))/
             (k*Abre[1])
  }

  if(rli == 0.0)
  {
    effli <- (k*Ab[1] - Ab[2]^2*r^2)/k
  }
  else
  {
    Abli <- ICoptk(k, rli, 1, 1, delta)
    effli <- (k*Ab[1] - Ab[2]^2*(r^2 - rli^2))/
             (k*Abli[1])
```


```
    }

    assign("eff", effre,
           envir=sys.frame(which = -2))
    return(effre-effli)
}

#Funktion zur Bestimmung des optimalen r0
#innerhalb eines bestimmten Bereichs [begin, end]
optimal <- function(k, start, ende)
{
    jetzt <- proc.time()[3]
    delta <- 1e-8
    eps <- 1e-6

    r <- uniroot(intervall, lower = start,
         upper = ende, tol = eps, maxiter = 50,
```


```
            k=k, rli=start, rre=ende,
            delta=delta)$root

    dauer <- proc.time()[3]-jetzt
    cat(round(r, 4), file="$PFAD/radius.txt")
    cat(round(eff, 4), file="$PFAD/ineff.txt")
    cat(round(dauer, 2), file="$PFAD/dauer.txt")

    return(c(r, eff))
}
```

Bemerkung: Man würde für diese Zwecke heute eher das Paket `CGIwithR` von David Firth verwenden, zu beziehen unter `http://cran.r-mirror.de/src/contrib/Descriptions/CGIwithR.html`.

8.4 Struktur von CRAN / das R Core Team

8.4.1 das CRAN

- CRAN steht für *C*omprehensive *R* *A*rchive *N*etwork
- $\hat{=}$ organisierte Verteilungsstruktur um R weltweit verfügbar zu machen
  - offizielle Mirrors http://cran.r-project.org/mirrors.html
  - verlinkt mit StatLib
  - weitere inoffizielle Mirrors: z.B.
    ftp://ftp.uni-bayreuth.de/pub/math/statlib/R

8.4.2 die R Foundation

8.4.2 (a) Was ist das?

- genauer Titel "*The R Foundation for Statistical Computing*"
- non–profit Organisation

- Sitz: in Wien, aktuell an der TU Wien
- eingetragene Vereinigung nach Österreichischem Recht

8.4.2 (b) Zielsetzung

- soll dem R–Projekt und anderen Innovationen im Bereich rechnergestützte Statistik Unterstützung gewähren
- soll eine Anlaufstelle sein für Privatleute, Instititutionen oder kommerzielle Firmen die R unterstützen wollen oder mit der R–Entwicklergemeinde in Kontakt treten wollen
- soll das Urheberrecht über die R–Software und –Dokumentation innehaben und verwalten
- vergleichbar zu Apache Foundation und GNOME Foundation
- Link zu den Statuten

8.4.2 (c) Leitung

- Präsidenten: Robert Gentleman, Ross Ihaka
- Generalsekretär: Friedrich Leisch
- Schatzmeister: Kurt Hornik
- Ehrenmitglied: John Chambers
- Auditoren: Peter Dalgaard und Martin Mächler

8.4.2 (d) Kontakt

- Adresse
  The R Foundation for Statistical Computing
  c/o Institut für Statistik und Wahrscheinlichkeitstheorie
  Technische Universität Wien
  Wiedner Hauptstraße 8-10/1071
  A-1040 Wien

Tel: (+43 1) 58801 10715
Fax: (+43 1) 58801 10798
Email: R-foundation@R-project.org

- Bankverbindung
  - Bank Austria Creditanstalt (Swift Code BKAUATWW)
  - innerhalb Österreich: Konto-Nr. 51582 121701, BLZ 12 000
  - international (IBAN): AT 93 1200 0515 8212 1701

8.4.2 (e) Mitgliedschaft

- gewöhnliche Mitgliedschaft
  - durch Mehrheitsvotum der Vollversammlung
  - Kriterium: nicht–monetäre Beiträge zur Entwicklung von R
- unterstützende Mitgliedschaft
  - durch Zahlung jährlicher Beiträge

- Mitgliedschaft sowohl natürlicher als auch juristischer Personen möglich
- Beiträge
  - natürliche Personen: 25 €
  - Institutionen: 250 €
  - Wohltäter ("benefactor") (?): 500 €

8.4.3 R Core Team

8.4.3 (a) Zusammensetzung / Organisation

- wie in Abschnitt 0.2.2 erwähnt:
  das "Projekt R" wurde initiiert von Ross Ihaka / Robert Gentlemen (University of Auckland)
- seit Mitte 1997 hat sich eine Kerngruppe von Entwicklern — das R Core Team zusammengefunden
- diese sind die einzigen mit Schreibrechten auf die R–Sourcen, wie sie sich im CRAN finden

- Zusammensetzung
  - Douglas Bates (USA): bates@stat.wisc.edu
  - John Chambers (USA): jmc@research.bell-labs.com
  - Peter Dalgaard (Dänemark): p.dalgaard@biostat.ku.dk
  - Robert Gentleman (USA): rgentlem@jimmy.dfci.harvard.edu
  - Kurt Hornik (Österreich): Kurt.Hornik@ci.tuwien.ac.at
  - Stefano Iacus (Italien): stefano.iacus@unimi.it
  - Ross Ihaka (Neuseeland): ihaka@stat.auckland.ac.nz
  - Friedrich Leisch (Österreich): Friedrich.Leisch@univie.ac.at
  - Thomas Lumley (USA): tlumley@u.washington.edu
  - Martin Mächler (Schweiz): maechler@stat.math.ethz.ch
  - Guido Masarotto (Italien): guido@hal.stat.unipd.it
  - Duncan Murdoch (Kanada): murdoch@stats.uwo.ca
  - Paul Murrell (NZ): paul@stat.auckland.ac.nz
  - Martyn Plummer (Frankreich): plummer@iarc.fr
  - Brian Ripley (Großbritannien): ripley@stats.ox.ac.uk

- Duncan Temple Lang (USA): duncan@research.bell-labs.com
- Luke Tierney (USA): luke@stat.umn.edu
- und — bis Oktober 1999: Heiner Schwarte (Schweiz) h.schwarte@bluewin.ch

8.4.3 (b) weitere wichtige Entwickler

- **Valerio Aimale (Italien)**: valerio.aimale@biosgroup.com **[*Entwickler der Command history; R–Tester in der* alpha*-Phase*]**
- **Thomas Baier (Österreich)**: Thomas.Baier@ci.tuwien.ac.at **[*Maintainer / Entwickler des* R-COM*–Servers*]**
- **Roger Bivand (Norwegen)**: Roger.Bivand@nhh.no **[*Maintainer / Entwickler der Pakete* GRASS, pixmap, spdep**]**
- **Ben Bolker (USA)**: ben@zoo.ufl.edu **[*Maintainer / Entwickler der Pakete* ape, gregmisc, landsc, boa, turtle, sparma, cannib, bbmisc**]**
- **Göran Broström (Schweden)**: gb@stat.umu.se **[*Maintainer / Entwickler der Pakete* eha, glmmML, spdep**]**
- **Vince Carey (USA)**: stvjc@channing.harvard.edu **[*Entwickler im Bioconductor Projekt; Themen: externe Referenzen und RDBMS in Bioinformatik; Maintainer / Entwickler der Pakete* combinat, gee, yags, outpack, cremo, Cmat, Cohort, alr, HIVresis, coild, lmsqreg**]**
- **Saikat DebRoy (USA)**: saikat@stat.wisc.edu **[*Thema: Konvertierung großer* R *Pakete auf* S4*–Klassen; Maintainer / Entwickler der Pakete* lme4, Matrix, nlme, RMySQL**]**
- **Lyndon Drake (Neuseeland)**: lyndon@stat.auckland.ac.nz **[*Maintainer / Entwickler des Paketes* gtkDevice**]**
- **Brian D'Urso (USA)**: durso@hussle.harvard.edu **[*LaTeX–*CM *Fonts für* R**]**

- **Dirk Eddelbüttel (Deutschland/USA)**: edd@debian.org **[*Maintainer / Entwickler des Paketes* RQuantLib, *zuständig für Debian Portierung*, *sehr aktiv in* R-help]**
- **John Fox (Kanada)**: jfox@mcmaster.ca **[*Autor zu Fox (2002); Maintainer / Entwickler der Pakete* car, effects, Rcmdr *(GUI für* R*)*, sem]**
- **Paul Gilbert (Kanada)**: pgilbert@bank-banque-canada.ca **[*Maintainer / Entwickler der Pakete* curve, dse1, dse2, dsepadi, juice, monitor, setRNG, syskern, tframe]**
- **Spencer Graves (?)**: spencer.graves@pdf.com **[*sehr aktiv in* R-help]**
- **Philippe Grosjean (Belgien/Frankreich)**: [der *Entwickler / Antreiber von GUI's für* R, *Entwickler von* SciViews; *c.f. auch hier*; *Maintainer der Rubrik auf der R–Homepage Entwickler der Paketes* pastecs, nlrq]
- **Julian Harris [*MacIntosh Portierung*]**
- **Torsten Hothorn (Erlangen)**: Torsten.Hothorn@rzmail.uni-erlangen.de **[*Maintainer / Entwickler der Paketes* StatDataML, exactRanktests, exactRanktests, ipred, lmtest, maxstat, multcomp, mvtnorm, RmSQL; *Thema: Bagging/Boosting usw.*]**
- **Robert King (Australien)**: robert.king@mailbox.gu.edu.au **[*Maintainer / Entwickler des Pakete* gld]**
- **Wing Kwong (Tiki) Wan [*MacIntosh Portierung*]**
- **Philippe Lambert (Belgien)**: lambert@stat.ucl.ac.be **[*Maintainer / Entwickler des Pakets* stable]**
- **Jan de Leeuw (USA)**: deleeuw@stat.ucla.edu **[*Maintainer / Entwickler des Pakete* homals]**
- **Andy Liaw (USA)**: andy_liaw@merck.com **[*Maintainer / Entwickler des Pakete* locfit, randomForest, spdep, *sehr aktiv in* R-help]**
- **Uwe Ligges (Dortmund)**: ligges@statistik.uni-dortmund.de **[*Maintainer / Entwickler des Pakets* scatterplot3D, *und der* winedt.ini *für* R, *siehe auch hier*; *sehr aktiv in* R-help]**
- **Jim Lindsey (Belgien)**: jlindsey@luc.ac.be **[*Maintainer / Entwickler der Pakete* gnlm, dna, event, repeated, growth, rmutil, stable, glim4; *R–Tester in der* alpha*-Phase*]**
- **Patrick Lindsey (Belgien)**: plindsey@luc.ac.be **[*Maintainer / Entwickler der Pakete* ordinal; *R–Tester in der* alpha*-Phase*]**

- **Catherine Loader (USA)**: catherine@research.bell-labs.com [***Maintainer / Entwickler des Pakets*** crossings, Locfit]
- **Gordon Maclean (USA)**: maclean@atd.ucar.edu [***R–Tester in der*** beta-***Phase***]
- **John Maindonald (Australien)**:john.maindonald@anu.edu.au [***Maintainer / Entwickler des Pakets*** hwde]
- **David Meyer (Österreich)**: david.meyer@ci.tuwien.ac.at [***Maintainer / Entwickler des Pakete*** StatDataML, vcd]
- **Steve Oncley (USA)**: oncley@atd.ucar.edu [***R–Tester in der*** beta-***Phase***]
- **Richard O'Keefe (Neuseeland)**: ok@cs.otago.ac.nz [***XML– und Datenstrukturen (Informatiker); aktiv in*** R-help]
- **Hubert Palme**: **(Wuppertal)** palme@uni-wuppertal.de [***R–Tester in der*** beta-***Phase***]
- **Jose Pinheiro (USA)**: jcp@research.bell-labs.com [***Autor von Pinheiro and Bates (2000); siehe auch hier; Maintainer / Entwickler der Pakete*** nlme, SASmixed]
- **Paulo J. Ribeiro, Jr. (Brasilien)**: Paulo.Ribeiro@est.ufpr.br [***Maintainer / Entwickler der Pakete*** geoR, geoRglm; ***siehe auch hier***]
- **Jonathan Rougier (Großbritannien)**: J.C.Rougier@durham.ac.uk [***Maintainer / Entwickler des Pakete*** 0array, Tensor]
- **Günther Sawitzki (Heidelberg)**: gs@statlib.uni-heidelberg.de [***Autor eines deutschsprachigen Tutorial, Entwickler für*** OBERON]
- **Martin Schlather (Bayreuth)**: martin.schlather@uni-bayreuth.de [***Maintainer / Entwickler des Pakets*** RandomFields]
- **Marc Schwartz (?)**: mschwartz@medanalytics.com [***Maintainer / Entwickler der Pakete*** X-Tab, ***Coauthor*** gregmisc; ***sehr aktiv in*** R-help]
- **Bill Simpson (Großbritannien)**: wsi@gcal.ac.uk [***Maintainer / Entwickler der Pakete*** GRASS, pixmap, spdep]
- **Gordon Smyth (Australien)**: smyth@wehi.edu.au [***Maintainer / Entwickler der Pakete*** StatMod, SMAWEHI, Tweedie; ***Maintainer der Domain*** StatSci.org]

- **Adrian Trapletti (Schweiz)**: adrian@olsen.ch **[*Maintainer / Entwickler der Pakete* tseries, ffnet]**
- **Terry Therneau (USA)**: therneau.terry@mayo.edu **[*Maintainer / Entwickler der Pakete* gamterms,plotterms, jitplot, mlowess, rpart, survival]**
- **Bill Venables (Australien)**: William.Venables@cmis.CSIRO.AU **[*Author von Venables and Ripley (1999) und Venables and Ripley (2000); wichtiger Programmierer für* S-Plus *und* R]**
- **Gregory R. Warnes (USA)**: warnes@biostat.washington.edu **[*Maintainer / Entwickler von* genetics, gregmisc, haplo.score, session]**
- **Andreas Weingessel (Österreich)**: Andreas.Weingessel@ci.tuwien.ac.at **[*Maintainer / Entwickler der Pakete* bindata, princurve, quadprog]**
- **Simon Wood (Großbritannien)**: snw@mcs.st-and.ac.uk **[*Maintainer / Entwickler der Pakete* mgcv, posum2]**
- **Achim Zeileis (Österreich)**: zeileis@ci.tuwien.ac.at **[*Mitorganisator der DSC–Konferenzen; Maintainer / Entwickler der Pakete* strucchange,lmtest, ineq, vcd]**

- Abstimmung der Entwicklungsaktivitäten über / bei
  - Konferenzen
  - Newsletter: R News
  - spezielles Forum: `r-devel` — `r-devel@stat.math.ethz.ch`
  - einheitliche, konsistente Entwicklung mit `subversion`–System — siehe auch

    http://developer.r-project.org/SVNtips.html

    BEISPIEL 8.4-1 [LOGFILE VOM 01.07.2003]:

```
Tue Jul 1 08:03:04 UTC 2003, maechler
  sapply() matrix: no dimnames if no ..
  R NEWS,1.1547.2.72
  R/src/library/base/R sapply.R,1.4.44.1
  R/src/library/base/man lapply.Rd,1.8.10.2
```


```
Tue Jul 1 08:03:37 UTC 2003, maechler
  branch update
  R NEWS,1.1669
  R/src/library/base/R sapply.R,1.5
  R/src/library/base/man lapply.Rd,1.12

Tue Jul 1 09:14:54 UTC 2003, ripley
  remove unnecessary weights.lm method
  simplify model.matrix.lm
  now model.matrix.default handles empty models
  effects() does not work on empty models
  R/src/library/base/man effects.Rd,1.7 lm.summaries.Rd,1.35
  R/src/library/base/R lm.R,1.106
  R/tests reg-tests-2.R,1.72 reg-tests-2.Rout.save,1.85
  R NEWS,1.1670

Tue Jul 1 09:39:37 UTC 2003, ripley
  lm.influence was misbehaving on a 0-rank model
  R/src/library/base/R lm.influence.R,1.15.2.1
  R NEWS,1.1547.2.73
```


```
Tue Jul 1 09:55:21 UTC 2003, hornik
  Updated.
  R FAQ,1.290.2.20
  R/doc/html faq.html,1.278.2.21
  R/doc/manual R-FAQ.texi,1.267.2.20
  R FAQ,1.310
  R/doc/html faq.html,1.298
  R/doc/manual R-FAQ.texi,1.287

Tue Jul 1 09:58:08 UTC 2003, hornik
  Typos and cosmetics.
  R NEWS,1.1671
  R/src/library/base/R library.R,1.127
  R/src/library/base/man library.dynam.Rd,1.16

Tue Jul 1 12:16:34 UTC 2003, duncan
  Fix from John to get  methods for [[ (and other primitives)
  merged into the methods table when a second or more library
  is loaded with such methods. See SWinTypeLibs and
  RDCOMClient for an example.
  R/src/library/methods/R RMethodUtils.R,1.47
```


```
Tue Jul 1 12:35:58 UTC 2003, ripley
  lm.influence -- handle 0-rank models, add names to
  components of return value
  R NEWS,1.1672
  R/src/library/base/R lm.influence.R,1.16
  R/src/library/base/man lm.influence.Rd,1.19
  R/tests reg-tests-2.R,1.73 reg-tests-2.Rout.save,1.86

Tue Jul 1 16:20:52 UTC 2003, ripley
  branch update
  R NEWS,1.1673

Tue Jul 1 22:03:58 UTC 2003, iacus
  R/src/modules/aqua aquaconsole.c,1.17
  R/src/unix aqua.c,1.5
  R/src/modules/aqua aquaconsole.c,1.18
  R/src/modules/aqua/Contents/Resources/main.nib info.nib,1.7
  objects.xib,1.9
  R/src/modules/aqua R.fix,1.3
```

- Bug–Tracking

– automatisierte Formulare zur Fehlermeldung, z.B.

http://r-bugs.biostat.ku.dk/cgi-bin/R

8.4.3 (d) Zuständigkeiten

- Douglas Bates: lineare Modelle, AIC, **prompt**, stack; Pakete `Devore5`, `Matrix`, `NISTNls`, `SASMixed`, `lme4`, `nlme`,
- John Chambers: "Vater" von `S` .... — vieles also direkt oder indirekt von ihm; Pakete `OOP`, `SLanguage`
- Peter Dalgaard: **as.function**, **edit.data.frame**, **gctorture**, **margin.table**, **match**.fun, **prop.table**, **subset**, **transform**; Pakete `TclTK`, `ISwR`
- Robert Gentleman: **match**.fun, toString, **match**.fun; Pakete `lgtdl`, `muhaz`, `panel`, `permax`
- Kurt Hornik: Editor von R–News, **aggregate**, agrep, **apropos**, **loglin**, **read.fwf** (`Perl`), [**r**,**p**,**q**,**d**]SignRank, [**r**,**p**,**q**,**d**]wilcox; Pakete `chron`, `date`, `mda`, `oz`, `polynom`, `tseries`, `vcd`, `chron`; aktuelle/ neue Projekte unter http://developer.r-project.org/TODO-KH.html
- Stefano Iacus: Paket `ifs`
- Ross Ihaka: Mitinitiator von R

- Friedrich Leisch: **formatC**, **mahalanobis**, RweaveLatex, Rtangle, Sweave, ...; Pakete `bindata`, `bootstrap`, `e1071`, `fracdiff`, `mlbench`, `multiv`, `pixmap`, `ifs`,
- Thomas Lumley: **close**[**make**,**read**].**socket**, esoph, **image**; Pakete `acepack`, `adapt`, `dichromat`, `leaps`, `netCDF`, `rmeta`, `survey`, `survival`
- Martin Mächler: **apropos**, Bessel, **check**.**options**, col2rgb, **example**, findInterval , **formatC**, **jitter** , **methods**, n2mfrow, **noquote**, **plot** . design, **plot** . **lm**, **print** . coefmat, **RNGkind**, str , **sunflowerplot**, **symnum**, **which**.**min**[**max**], xyz . **coords**, **apropos**; Pakete `VLMC`, `cluster`, `cobs`, `lasso2`, `lokern`, `lpridge`, `normix`, `wavethresh`; aktuelle/ neue Projekte unter `http://developer.r-project.org/TODO-MM.html`
- Guido Masarotto: **library** , **link** . html.**help**, png, Rwin configuration [Konfiguration R für Windows], saveplot , **system**, windows,
- Duncan Murdoch: Paket `ellipse`
- Paul Murrell: **layout**, plotmath, Paket `grid`; aktuelle/ neue Projekte unter `http://developer.r-project.org/paul-todo.html`
- Martyn Plummer: Paket `gtkDevice`
- Brian Ripley: " R wie Ripley ....", extrem aktiv in `r-help`; **add1**, **alias** , **aov**, bandwidth, **C**, **chull**, **conflicts** , ddebugger, **dev2bitmap**, **dummy**.**coef**,

eff.aovlist, **expand.grid**, **extractAIC**, **factor**.scope, **file**.access, **file**.info, files, **kappa**, **labels**, **link**.html.**help**, **list**.files, manova, **max.col**, **model**.tables, **page**, **pmatch**, png, **poly**, **predict**.**glm**, **proj**, RANDOM, **read.fwf**, **relevel**, repilcations, **rug**, savePlot, **se**.**aov**, **se**.contras, sets, shell, **step**, **summary**.manova, Sys.info, Sys.sleep, **system**, **ts**−**methods**, **update**, windows, **zip**.**file**.**extract**; Pakete `KernSmooth`, `MASS`, `RODBC`, `boot`, `class`, `gee`, `logspline`, `mix`, `nnet`, `pspline`, `rpart`, `sm`, `spatial`, `tree`, `KernSmooth`; aktuelle/ neue Projekte unter http://developer.r-project.org/BDR-TODO.html

- Duncan Temple Lang: [Programmierung; multiple threading...], getNativeSymbolInfo, getNumCConverters, getNumCConverters; Pakete `REventLoop`, `RGdkPixbuf`, `RGtkxxx`–Pakete, `RObjectTables`, `RSPerl`, `SASXML`, `SJAVA`, `REventLoop`, `RWinRegistry`, `RWinTypeLibs`, `SXalan`, `Slcc`, `Sxslt`, `XML`, `REventLoop`; aktuelle/ neue Projekte unter http://developer.r-project.org/TODO-DTL.html
- Luke Tierney: viele konzeptionelle Ideen, u.a. *Namespaces*; siehe auch Beiträge auf http://developer.R-project.org/; bindenv, ns−xxx, bindenv; Pakete `serialize`, `snow`, `tkrplot`
- Grafiktreiber c.f. http://www.stat.auckland.ac.nz/~paul/R/devices.html

Treiber	Betriebssystem	zuständig
PostScript (und bitmap)	alle	R-core
PicTeX	alle	R-core
PDF	alle	R-core
xfig	alle	R-core
Java	alle	Duncan Temple-Lang
GTK	alle	Martyn Plummer
SVG	alle	T Jake Luciani
X11 (und PNG und JPEG)	*NIX	R-core
GNOME	*NIX	Martyn Plummer
Quartz	*NIX	Stefano Iacus
Windows	Windows	Duncan Murdoch
proxy	Windows	Thomas Baier
Macintosh	Macintosh	Stefano Iacus

8.4.4 Einreichung eigener Pakete bei CRAN

8.4.4 (a) Vorbereitungen

- Anlegen eines Pakets — vgl. Abschnitt 8.2.2
- Bereinigen und Profilierung des Codes — vgl. Abschnitt 8.3.2
- **systematische Dokumentation des Codes und der Daten**
  - siehe auch Abschnitte 3.3.4 und 8.2.6
  - Referenz: Writing R Extensions, Kapitel 2

8.4.4 (b) Einreichen bei CRAN

- packen und zippen mit `tar -cf` und `gzip` auf ein File mit Namen `<name>.tar.gz`
- dabei muss `name` von der Form sein
  `<Paketname>_version[_engine[_type]]` mit:
  - `[....]` ist optional

  - `<Paketname>` und `version` sind konsistent mit den Einträgen in `DESCRIPTION` File
  - `engine` gibt die `S`–Implementation an, für die das Paket gedacht ist — per default R
  - `type` gibt an, ob es sich um Quellen oder Binärdateien für bestimmte Systeme handelt — siehe Writing R Extensions
  - Idee: Nutzbarkeit von `S-Plus`–Code auch für R
- upload auf CRAN mit `ftp` auf `ftp://ftp.ci.tuwien.at/incoming`
- Referenz: Writing R Extensions (2006b, Kapitel 1.5)
- formale Voraussetzungen:
  - **R** CMD **check** muss fehlerfrei laufen — vgl. Abschnitt 8.2.11
  - **notwendig**: License Statement in `DESCRIPTION`

  ⇝ Feedback von allen möglichen Nutzern, ob das Paket auf allen möglichen esoterischen Konfigurationen läuft

Aufgabensammlung zur Veranstaltung

R/S-Plus für Einsteiger und für Fortgeschrittene

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

y

A Aufgaben

A.1 Blatt 1

- Stoff bis einschließlich Abschnitt 1.2.2 — Vektoren, Matrizen, Arrays

A.1.1 Arbeit mit R-Skripten am Beispiel eines beliebigen Editors: (ohne direkte Anbindung an R)

(a) Erzeugen Sie eine R-Datei.

(b) Laden und arbeiten Sie die komplette (gespeicherte) Datei ab.

(c) Laden und arbeiten Sie die ersten m Befehle ab.

(d) Laden und arbeiten Sie einzelne Befehle ab.

Lösungsvorschlag:

- `.pdf`-Version: L.1.1

A.1.2 Auffinden von Datensätzen

(a) Suchen Sie im Internet einen Datensatz, in dem die tägliche Kursentwicklung von Dollar zu DM/Euro über einen längeren Zeitraum dargestellt ist.

(b) Suchen Sie im Internet einen Datensatz, in dem Klimadaten einer Wetterstation in Deutschland dargestellt sind.

(c) Welche Pakete (packages) sind in `S-Plus/R` integriert? Wie lassen sich diese laden?

Lösungsvorschlag:

- `.pdf`-Version: L.1.2

A.1.3 Auffinden von Datensätzen II — ohne Lösung

:

(a) Suchen Sie im Internet einen Datensatz, in dem die Bevölkerungsentwicklung der Bundesrepublik Deutschland seit 1945 dargestellt ist. Schicken Sie die entsprechenden Links per E-Mail an den Kursleiter.

(b) Suchen Sie nach einem Datensatz, in dem das Brutto-Inlandsprodukt (GNP) der USA in den Jahren seit 1945 dargestellt ist. Schicken Sie die entsprechenden Links per E-Mail an den Kursleiter.

(c) Welche großen sechs (`.pdf`-)Manuals werden mit R mitvertrieben?

A.1.4 Datenimport

(a) Welche Funktionen eignen sich zum Einlesen von Daten?

(b) Importieren Sie einen Beispieldatensatz (z.B. Währungs- oder Klimadaten aus Aufgabe A.1.2).

Lösungsvorschlag:

- `.pdf`-Version: L.1.4
- `.R`-Version (local): Blatt 1, Aufgabe 4(c)
- `.R`-Version (www): Blatt 1, Aufgabe 4(c)

A.1.5 Mustererzeugung

Erzeugen Sie eine Matrix M mit 20 Spalten und den folgenden Zeileneinträgen:

Zeile 1: 1, 2, ..., 20

Zeile 2: Zahlen zwischen 0.25 und 5 mit Abstand 0.25

Zeile 3: Spalte 1-10 mit Eintrag 1, Spalte 11-20 mit Eintrag 2

Zeile 4: 1, 1, 2, 2, 1, 1, 2, 2, ...

Zeile 5: 1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6

Zeile 6: 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3

Lösungsvorschlag:

- `.pdf`-Version: L.1.5
- `.R`-Version: Blatt 1, Aufgabe 5
- `.R`-Version (www): Blatt 1, Aufgabe 5

A.1.6 Mustererzeugung II — ohne Lösung

Erzeugen Sie eine Matrix M mit 20 Spalten und den folgenden Zeileneinträgen:

Zeile 1: Zahlen zwischen 0.5 und 10 mit Abstand 0.5

Zeile 2: Spalte 1-10 mit Eintrag 3, Spalte 11-20 mit Eintrag 0

Zeile 3: 1, 2, 1, 2, 1, 2, 1, 2, ...

Zeile 4: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6

Zeile 5: 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, 0, 0, 0, 0, 0, 0

Zeile 6: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Zeile 7: 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 8

A.2 Blatt 2

- Stoff bis einschließlich Abschnitt 1.9 — Datenausgabe

A.2.1 Indexoperationen, Matrizen

(a) Erzeugen Sie die folgenden Matrizen:

$$\begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \quad \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 9 & 16 & 25 & 36 & 49 \\ 8 & 27 & & & & \\ & & \vdots & \vdots & \vdots & \vdots \\ 16 & 81 & \vdots & \vdots & \vdots & \vdots \\ 32 & 243 & & & & \end{pmatrix}$$

Hinweis: Denken Sie an logische Indexoperationen.

(b) Laden Sie den `painters`-Datensatz aus des MASS-Pakets. Finden Sie alle Malernamen, die mindestens 3 mal den Buchstaben "e" enthalten.

Lösungsvorschlag:

- `.pdf`-Version: L.2.1
- `.R`-Version: Blatt 2, Aufgabe 1
- `.R`-Version (www): Blatt 2, Aufgabe 1

A.2.2 Indexoperationen, Matrizen II —ohne Lösung

(a) Erzeugen Sie die folgenden Matrizen:

$$\begin{pmatrix} 6 & 5 & 4 & 3 & 2 & 1 \\ 0 & 6 & 5 & 4 & 3 & 2 \\ 0 & 0 & 6 & 5 & 4 & 3 \\ 0 & 0 & 0 & 6 & 5 & 4 \\ 0 & 0 & 0 & 0 & 6 & 5 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{pmatrix} \quad \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 2 & 0 & 0 & 0 \\ 0 & 2 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 2 & 0 \\ 0 & 0 & 0 & 2 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 4 & 8 & 16 & 32 \\ 2 & 4 & 8 & 16 & 32 & 64 \\ 4 & 8 & 16 & 32 & 64 & 128 \\ 8 & 16 & & & & \\ & & \vdots & \vdots & \vdots & \vdots \\ 16 & 32 & & & & \\ 32 & 64 & & & & \end{pmatrix}$$

(b) Laden Sie den `painters`-Datensatz aus des MASS-Pakets. Finden Sie alle Malernamen, die höchstens 2 mal den Buchstaben “e” enthalten.

A.2.3 Faktoren

(a) Laden Sie den `iris`-Datensatz. Geben Sie dann für die einzelnen `Species` jeweils das Minimum und das Maximum für `Sepal.Length` und `Sepal.Width` an.

(b) Geben Sie für die Art `setosa` an, welche Werte der Variable `Sepal.Length` bzw. `Sepal.Width` jeweils am häufigsten angenommen werden.

Lösungsvorschlag:

- `.pdf`-Version: L.2.3
- `.R`-Version: Blatt 2, Aufgabe 3
- `.R`-Version (www): Blatt 2, Aufgabe 3

A.2.4 Einladen und Umformatieren von Datensätzen aus dem Netz

(a) Beziehen Sie unter `http://de.finance.yahoo.com` die Tagesschlusskurse von BMW (`BMW.DE`) und DaimlerChrysler (`DCX.DE`) vom 16.05.2005 bis 16.05.2006.

(b) Analysieren Sie die entsprechende URL mit dem Query-String in Ihrem Browser. Schreiben Sie eine Funktion, die zu gegebenem Zeitraum `dd.mm.yyyy-dd.mm.yyyy` (bzw. den Größen `d1`, `m1`, `y1`, `d2`, `m2`, `y2`) und Wertpapierkennung (wie `BMW.DE`) die entsprechenden Tagesschlusskurse aus dem Internet mit `download.file` herunterlädt und in einen vorgegebenen Dataframe liest.

Lösungsvorschlag:

- `.pdf`-Version: L.2.4
- `.R`-Version: Blatt 2, Aufgabe 4
- `.R`-Version (www): Blatt 2, Aufgabe 4

A.2.5 String-, Matrixoperationen

Weisen Sie eine der Matrizen aus Aufgabe A.2.1 einem Element mit Namen *Hilbert* einer Liste `L` zu und geben Sie deren Eigenwerte in Form einer Ausgabe vom Typ

`Die Eigenwerte der Matrix` *Hilbert* `sind` *1.00, 2.00, 3.24, ...*

Die kursiv geschriebenen Ausdrücke beziehen Sie dabei direkt von `L` und formatieren Sie die Eigenwerte wie angedeutet.
(Hinweis: **formatC**)

Lösungsvorschlag:

- `.pdf`-Version: L.2.5
- `.R`-Version: Blatt 2, Aufgabe 5
- `.R`-Version (www): Blatt 2, Aufgabe 5

A.3 Blatt 3

- Stoff bis einschließlich Abschnitt 1.9 — Datenausgabe (Schwerpunkt: Umgang mit Zeichenketten)

A.3.1 Umgang mit Zeichenketten

Laden Sie (ohne R) unter http://www.mc-rall.de/bibschlt.zip

> "*Die Heilige Schrift des Alten und Neuen Testaments*",
> nach dem Urtext übersetzt von Franz Eugen Schlachter.
> Neue Überarbeitung 1951 durch Genfer Bibelgesellschaft.

als gezippte ASCII-Datei herunter und entpacken Sie diese. Mit dem Befehl

paste(**scan**("bibschl.txt",what=**character**(0)),sep="␣") laden Sie den Text in eine Variable und machen Sie mit dieser Variablen folgende Analysen:

(a) Wieviele Worte (inkl. Sonderzeichen) enthält der Text?

(b) Eliminieren Sie alle Steuerungszeichen ([: cntrl :]) und Satzzeichen ([: punct :]) — vgl. ?regexp.

(c) Zählen Sie, wie oft der Name "Jesus" (auch als "Jesu", "Jesum") vorkommt, wie oft der Name "Menschensohn".

(d) Verwenden Sie **unique**, um Doppeltnennungen im Ergebnis (b) zu eliminieren. Wieviele verschiedene Wörter sind das?

(e) Ersetzen Sie im ursprünglichen Text sämtliche Umlaute "Ä,ä,ö,Ö,Ü,ü,ß" durch "Ae,ae,oe,Oe,Ue,ue,ss".

(f) Finden Sie aus dem Ergebnis von (d) alle Wörter, die mindestens zwei "j" (egal ob "j" oder "J") enthalten.

(g) Finden Sie alle Worte, die mindestens ein "j" (egal ob "j" oder "J") aber kein "e" enthalten.

(h) Finden Sie alle Worte, die mindestens 5 Zeichen lang sind, ein "j" (kein "J") und ein "u" (egal ob "u" oder "U") enthalten.

Lösungsvorschlag:

- `.pdf`-Version: L.3.1
- `.R`-Version: Blatt 3, Aufgabe 1
- `.R`-Version (www): Blatt 3, Aufgabe 1

A.3.2 Umgang mit Zeichenketten II

Verwenden Sie wieder den Datensatz "`bibschl.txt`" aus Aufgabe A.3.1.

(a) Erstellen Sie mit **sapply**, **length**, **grep** und der vorgegebenen Variable LETTERS eine Tabelle, in der Sie zu jedem Buchstaben (ohne Groß/Kleinschreibung zu beachten) feststellen, in wievielen Wörtern in der Bibel er vorkommt. Wenn Sie wollen, können Sie dies auch mit den Sonderzeichen tun.

(b) Finden Sie mit **sapply**, tolower und **nchar** heraus, wie lang das längste Wort (ohne Satzzeichen, Steuerzeichen) ist. Erstellen Sie eine Aufstellung mit den zehn längsten Wörtern (ohne Mehrfachnennungen, und Groß/Kleinschreibung zu beachten).

(c) Filtern Sie mit **grep** alle Zahlwörter heraus und erstellen Sie erneut eine Aufstellung mit den zehn längsten Wörtern (ohne Mehrfachnennungen, und Groß/Kleinschreibung zu beachten).

(d) Erstellen Sie eine Häufigkeitsaufstellung (**table**) der Wortlängen (ohne Groß/Kleinschreibung zu beachten).

(e) Erstellen Sie eine Häufigkeitsaufstellung der Wörter (ohne Groß/Kleinschreibung zu beachten) Geben Sie die zehn häufigsten Wörter aus (**sort** auf das Ergebnis von **table**).

Lösungsvorschlag:

- `.pdf`-Version: L.3.2
- `.R`-Version: Blatt 3, Aufgabe 2
- `.R`-Version (www): Blatt 3, Aufgabe 2

A.3.3 Umgang mit regulären Ausdrücken / regexp's

Laden Sie mit `download.file` die Serviceseite `http://www.uni-bayreuth.de/departments/math/org/mathe7/services.html` herunter. Filtern Sie anschließend alle URLS, die in HTML-Ausdrücken von der Form "`< a href ....>`" eingeschlossen sind, in einen Vektor `Links` (ohne HTML-tag "`< a href ....>`").

Stichworte: `grep`, `strsplit`, `scan` mit `what=character(0)`, `paste`, `gsub`.

Lösungsvorschlag:

- `.pdf`-Version: L.3.3
- `.R`-Version: Blatt 3, Aufgabe 3
- `.R`-Version (www): Blatt 3, Aufgabe 3

A.3.4 Matrixoperationen

(a) Erzeugen Sie die Matrix

$$M = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/2 & 1/3 & & \dots & & 1/7 \\ & & \ddots & & & \\ 1/3 & & & & & \\ & \vdots & & & & \vdots \\ 1/4 & & & & & \\ 1/5 & & & & & \\ 1/6 & 1/7 & & \dots & & 1/11 \end{pmatrix}$$

(b) Bestimmen Sie die Determinante und Kondition von M.

(c) Lösen Sie das lineare Gleichungssystem $Mx = (1, 2, 3, 4, 5, 6)^T$.

Verhältnis von Betrag des größten zum Betrag des kleinsten Eigenwerts; ist die Kondition gross, so spricht man von einer schlecht konditionierten Matrix

Lösungsvorschlag:

- `.pdf`-Version: L.3.4
- `.R`-Version: Blatt 3, Aufgabe 4
- `.R`-Version (www): Blatt 3, Aufgabe 4

A.3.5 Schreiben von Daten auf File

Schreiben Sie das Ergebnis aus Aufgabe A.2.5 auf 3 Stellen nach dem Komma formatiert in ein File "`erg.txt`".

Lösungsvorschlag:

- `.pdf`-Version: L.3.5
- `.R`-Version: Blatt 3, Aufgabe 5
- `.R`-Version (www): Blatt 3, Aufgabe 5

A.3.6 Schreiben von Daten auf File II — ohne Lösung

Schreiben Sie den in Aufgabe A.2.4 erzeugten Dataframe (Sie dürfen die Musterlösung verwenden) auf File

(a) — im `.Rdata`–Format mit **save**

(b) — als ASCII-Datei das File `"Aktienkurs.dat"`, und zwar zeilenweise im Format:

```
<tt> <mm> <jjjj> <BMW> <DAIMLER>
```

Dabei sollen `<tt>` und `<mm>` ganze Zahlen mit führender 0 sein und `<BMW>`, `<DAIMLER>` jeweils auf zwei Nachkommastellen genau sein. Außerdem sollen alle zehn Zeilen eine Freizeile eingefügt werden, sowie eine Kommentar-Titelzeile über den eigentlichen Daten. Die Einträge in einer Zeile sollen jeweils durch einen Tabulator getrennt werden.

A.4 Blatt 4

- Stoff bis einschließlich Abschnitt 2.4.7 — getrimmte und winsorisierte Mittel
  Schwerpunkt: empirische Datenanalyse

A.4.1 Skalenniveaus

Klassifizieren Sie die folgenden Merkmale und begründen Sie jeweils, ob Mittelwert, Median und Modus für diese Merkmale zulässig bzw. sinnvoll sind

- Geschlecht
- Schulnoten
- Semesteranzahl
- Ideale Normalverteilung

Lösungsvorschlag:

- `.pdf`-Version: L.4.1

A.4.2 Skalenniveaus II — ohne Lösung

Aus einem (leicht verfremdeten) Beratungsfall:
Eine Firma vertreibt Fäden. Es geht darum festzustellen, welche Faktoren die Qualität der Fäden beeinflusst.
Folgende Informationen stehen zur Verfügung

- "Faden"
  - Charge: Materialschwankungen zw. 6 verschiedene Lieferungs-Chargen
  - Material: Unterschiede in Ausgangsmaterialien: 3 Fadentypen (S, A, B)
  - Durchmesser
    - P1 = 0.05 mm - 0.10 mm,
    - P2 = 0.20 mm - 0.25 mm
- "Verpackungsprozess"
  - Fließbandgeschwindigkeit
  - Arbeiter — wer hat das Verpacken beaufsichtigt?
  - Rollenart: Rolle A = weich usw. bis Rolle D

Überlegen Sie sich jeweils, welche Skalenniveaus (kategorial, ordinal, metrisch (intervall-/ratio-skaliert)) für diese Informationen angebracht sind.

A.4.3 Univariate Analyse

Beziehen Sie auf der Service–Homepage oder unter

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/kredit1.txt

den Datensatz `kredit1.txt`:

(a) Analysieren Sie separat die Variablen *Laufzeit*, *Zahlungsmoral* und *Kredithöhe* des `kredit1` Datensatzes. Beachten Sie dabei die verschieden Skalenniveaus.

(b) Verwenden Sie **boxplot**() und **hist** () für diejenigen Variablen, für die dies möglich/sinnvoll ist.

Lösungsvorschlag:

- `.pdf`-Version: L.4.3
- `.R`-Version: Blatt 4, Aufgabe 3
- `.R`-Version (www): Blatt 4, Aufgabe 3

A.4.4 Elementare Datenanalyse

Beziehen Sie auf der Service–Homepage oder unter

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/baby.txt

den Datensatz `baby.txt`:

> Von 32 Babies wurde die Masse zur Geburt M und der Massezuwachs (in % der Geburtsmasse) innerhalb des 70sten bis 100sten Tages nach der Geburt R registriert (Daten nach P. Armitage (1971): *The Theory of Linear Models and Multivariate Analysis.* New York: Wiley.). Die Werte liegen als `ASCII`–Text vor (incl. 2 Kopfzeilen), und wir wissen, daß es sich um 32 Zeilen mit 2 Spalten handelt.

(a) Lesen Sie die Daten in ein Array / eine Matrix der Dimension $32, 2$.

(b) Geben Sie ein Histogramm von M aus.

(c) Berechnen Sie Mittelwert, Median, Modalwert, Varianz und Standardabweichung von M, sowie die Korrelation von M und R — alle jeweils auf Basis der empirischen Verteilung. Interpretieren Sie die Ergebnisse.

(d) Erstellen Sie zu R einen Boxplot und interpretieren Sie diesen.

Lösungsvorschlag:

- `.pdf`-Version: L.4.4
- `.R`-Version: Blatt 4, Aufgabe 4
- `.R`-Version (www): Blatt 4, Aufgabe 4

A.4.5 Elementare Datenanalyse II

(a) Beziehen Sie mit der Musterlösung zu Aufgabe A.2.4 wieder die Tagesschlusskurse von BMW (`BMW.DE`) und DaimlerChrysler (`DCX.DE`) des letzten Jahres, genauer vom 22.05.2005 bis 22.05.2006.

(b) Schreiben Sie diese in einen vorgegebenen Dataframe mit Spaltennamen `"Daimler"` und `"BMW"`.

(c) Gehen Sie von den Tagesschlusskursen X_i jeweils über zu den relativen Tagesänderungen $X_i/X_{i-1} - 1$. Vergleichen Sie die Performances der beiden Aktien graphisch mit einem Boxplot.

(d) Berechnen Sie die (empirische) Korrelation der relativen Tagesänderungen der beiden Aktien.

Lösungsvorschlag:

- `.pdf`-Version: L.4.5
- `.R`-Version: Blatt 4, Aufgabe 5
- `.R`-Version (www): Blatt 4, Aufgabe 5

A.5 Blatt 5

- Stoff bis einschließlich Abschnitt 2.4.7 — getrimmte und winsorisierte Mittel
  Schwerpunkt: elementare Stochastik

A.5.1 Univariate Konvexkombinationen

(a) Simulieren Sie $n = 100$ Realisationen einer univariaten Zufallsvariable X, welche der folgenden Verteilung folgt

$$\mathcal{L}(X) = 0.9 \cdot \mathcal{N}(0, 1) + 0.1 \cdot \mathcal{N}(3, 1)$$

Interpretieren Sie $\mathcal{L}(X)$ wie folgt: mit Wahrscheinlichkeit 90% stammen die Daten aus einer idealen Situation, $\mathcal{N}(0, 1)$, und mit Wahrscheinlichkeit 10% handelt es sich um Ausreißer, die aus $\mathcal{N}(3, 1)$ stammen.

(b) Analysieren Sie die Stichprobe aus Teil (a) mit Hilfe von **summary()** and **stem()** und interpretieren Sie die Resultate.

(c) Erzeugen Sie einen Boxplot der Stichprobe aus Teil (a) und interpretieren Sie diesen.

Lösungsvorschlag:

- `.pdf`-Version: L.5.1
- `.R`-Version: Blatt 5, Aufgabe 1
- `.R`-Version (www): Blatt 5, Aufgabe 1

A.5.2 Univariate Konvexkombinationen II —ohne Lösung

Sie wollen sich klar machen, wie gut der klassische Schätzer S_n für die Standardabweichung $S_n^2 = \frac{1}{n-1}\sum_{j=1}^{n}(x_i - \bar{x}_n)^2$, in R realisiert durch **sd**, für Beobachtungen $X_i \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\mu, \sigma^2)$ im Vergleich zum MAD, in R realisiert durch **mad**, ist.

(a) Erzeugen Sie für $n = 5, 10, 100$ jeweils $M = 10000$ (Pseudo)Zufallszahlen $X_{i,j} \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0, 1)$, $i = 1, \ldots, n$, $j = 1, \ldots, M$ und schreiben Sie das Ergebnis in eine $M \times n$ Matrix.

(b) Berechnen Sie jeweils für die M Stichproben mit **sd** und **mad** Schätzungen $S_j^{(\text{sd})}$ und $S_j^{(\text{mad})}$ für $\sigma = 1$. Schreiben Sie die Ergebnisse in einen Dataframe erg der Dimension $M \times 2$ und stellen Sie die Ergebnisse in Boxplots dar.

(c) Berechnen Sie jeweils einen Schätzwert für den mit n standardisierten mittleren quadratischen Fehler

$\mathrm{nMSE} = n\mathrm{MSE}(S^{(\mathrm{mad/sd})}) = n\,\mathrm{E}[(S^{(\mathrm{mad/sd})} - 1)^2]$ durch Ausdrücke der Art n∗**mean**(erg[,1]−1)^2

(d) Erzeugen Sie nun wieder für $n = 5, 10, 100$ jeweils $M = 10000$ (Pseudo)Zufallszahlen
$Y_{i,j} \overset{\text{u.i.v.}}{\sim} (1 - \frac{r}{\sqrt{n}})\mathcal{N}(0,1) + \frac{r}{\sqrt{n}}\,\mathrm{Dirac}(25)$, $i = 1, \ldots, n$, $j = 1, \ldots, M$ für $r = 0.5$ (siehe auch Aufgabe A.4.5 in den Aufgaben im R-Kurs-Manuskript).

(e) Wiederholen Sie Teil (b) und (c) nur jetzt mit den Y- statt der X-Variablen. Welchen Schluss ziehen Sie aus Ihren Ergebnissen?

A.5.3 Verteilungen in R — ohne Lösung

Berechnen/erzeugen Sie

(a) den Median einer Poissonverteilung mit Parameter $\lambda = 2$

(b) den Interquartilsabstand einer Exponentialverteilung mit Parameter $\lambda = 2$ (also "oberes − unteres Quartil")

(c) die Wahrscheinlichkeit, dass eine $\mathcal{N}(2, 4)$-verteilte Zufallsgröße zwischen 4 und 6 liegt.

(d) die Dichte der F-Verteilung mit Nichtzentralitätsparameter 2 und 4 und 7 Freiheitsgraden

(e) 1000 $\mathrm{Geom}(1/6)$-Variablen

(f) (jeweils) die Wurzel aus den Variablen aus (e)

(g) die Wahrscheinlichkeit, dass bei 5maligem Ziehen ohne Zurücklegen aus einer Urne mit 13 weißen und 7 roten Kugeln 4 rote herauskommen.

(h) die Wahrscheinlichkeit, dass bei 5maligem Ziehen mit Zurücklegen aus einer Urne mit 13 weißen und 7 roten Kugeln 4 rote herauskommen.

(i) das obere 5%-Quantil einer χ^2-Verteilung mit 7 Freiheitsgraden und Nichtzentralität 4

(j) die Wahrscheinlichkeit, dass die Wurzel aus einer Poisson($\lambda = 3$) Variable nicht größer ist als 4

(k) die Wahrscheinlichkeit, eine uniform $[0, 1]$-verteilte Variable nicht größer ist als 0.4

(l) 35 Würfelwürfe (`sample` mit Argument `replace=TRUE`).

(m) die Dichte der Gamma-Verteilung mit Parametern $\lambda = 2$ und $\mathtt{shape} = 2$ an den Stellen 0 bis 1 (Schrittweite 0.01)

(n) die Dichte der t-Verteilung mit Nichtzentralitätsparameter 2 und 4 Freiheitsgraden an der Stelle 0.01

(o) die Wahrscheinlichkeit, dass ich öfter als 25 mal beim Roulette

auf schwarz setzen muss, um zum 10ten mal zu gewinnen

(p) die Wahrscheinlichkeit, die die Beta-Verteilung mit Parametern 3 und 2 auf den Bereich zwischen 0.4 und 0.6 wirft.

A.5.4 Umgang mit Zusatzpaketen `distr` / `distrEx`

(a) Berechnen Sie mit `distrEx` den theoretischen Erwartungswert, die theoretische Standardabweichung und den theoretischen Median der Verteilungen von X_1, X_2, X_3 wobei $X_1 \sim \mathrm{Pois}(0.3)$, $X_2 \sim t_7(0)$ und $X_3 = \log(1 + \exp(X_1 + 3.5X_2))$.

(b) Schreiben Sie mit den Paketen `distr` und `distrEx` eine Funktion zur Visualisierung des Zentralen Grenzwertsatzes: Übergabeparameter sei eine Variable `Vert` vom Typ AbscontDistribution, also einer Verteilung mit Dichte, und eine Zahl n. Ihre Funktion soll die Dichte der Verteilung der standardisierten und zentrierten Summe $S_n^\natural = (S_n/n - \mathrm{E}[X_1])/\sqrt{\mathrm{Var}[X_1]/n}$ für $S_n = \sum_{i=1}^n X_i$ auf dem Gitter x=**seq**(−4,4,0.01) gegen die Dichte von $\mathcal{N}(0,1)$ plotten (Stichwort **matplot**(type="l"), wobei $X_i \overset{\text{u.i.v.}}{\sim}$ `Vert`. Testen Sie die Funktion mit Vert=Exp(1), Vert=**sin**(Exp(1)) und n=**c**(1:10,20); zur Erzeugung von S_n dürfen Sie hier der

Einfachheit halber auch folgenden Code benutzen:

```
i ← 0; Sn ← 0
while (i<n) {i ← i+1; Sn ← Sn+X }
```

Lösungsvorschlag:

- .pdf-Version: L.5.4
- .R-Version: Blatt 5, Aufgabe 4
- .R-Version (www): Blatt 5, Aufgabe 4

A.5.5 Übungsaufgaben zur Stochastik

(a) Berechnen Sie die Zahl N_0 an Personen in einem Raum, ab der die Wahrscheinlichkeit größer ist als 25%, dass mindestens zwei Personen am gleichen Tag Geburtstag haben — denken Sie an **qbinom**.

(b) Sie wollen feststellen, wieviele Reiskörner (N) in einem großen Gefäß sind. Dazu entnehmen Sie dem Gefäß eine Handvoll Reis, zählen dort die Körner (ergibt $n = 120$), färben Sie rot und geben Sie wieder in das Gefäß. Anschließend schütteln Sie das Gefäß kräftig und ziehen wieder eine Handvoll Reis, zählen die Körner — $m = 100$ — und stellen fest, wieviele davon rot sind — $k = 23$. Geben Sie ein $\hat{N}$ an, so dass die Wahrscheinlichkeit, dass $\hat{N} < N$ ist, kleiner ist als 5% — denken Sie an **dhyper**.

(c) Sie waschen $N = 20$ Paar schwarze Socken. Diese sind auf den ersten Blick nicht zu unterscheiden, bei genauerer Untersuchung aber schon. Daher ziehen Sie "blind" ohne Zurücklegen aus der

Trommel. Ab wieviel (einzelnen) Socken ist die Wahrscheinlichkeit größer als 70%, dass Sie mindestens ein Paar beisammen haben?

Lösungsvorschlag:

- `.pdf`-Version: L.5.5
- `.R`-Version: Blatt 5, Aufgabe 5
- `.R`-Version (www): Blatt 5, Aufgabe 5

A.6 Blatt 6

- Stoff bis einschließlich Abschnitt 2.9.3 — Bootstrap

A.6.1 Visualisierung des (schwachen) Gesetzes der großen Zahlen

(a) Erzeugen Sie für $M = 300$ und $N = 1, 3, 5, 10, 50, 100, 1000$ u.i.v. (Pseudo–)–Zufallsgrößen $\omega_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, M$ mit $P(\omega_{i,k} = j) = \frac{1}{6}$ für $j = 1, \ldots, 6$, und daraus die abgeleitete Größen $s_{i,k} := \mathrm{I}_{\{\omega_{i,k}=6\}}$. Welche Interpretation besitzen die $s_{i,k}$?

(b) Berechnen Sie die im Gesetz der großen Zahlen auftauchenden Größen $S_{k,N} := \frac{1}{N} \sum_i s_{i,k}$. Warum berechnen wir hier M verschiedene Werte $S_{k,N}$?

(c) Erzeugen Sie für die verschiedenen N jeweils ein Histogramm und eine (empirische) Verteilungsfunktion der $S_{k,N}$. Zur komprimierten Visualisierung können Sie auch Boxplots verwenden.

(d) Berechnen Sie die (empirische) Varianz V_N der $S_{k,N}$ in

Abhängigkeit von N,

$$V_N := \frac{1}{M} \sum_k (S_{k,N} - \frac{1}{6})^2$$

In welcher Rate schrumpft die Varianz — und damit in welcher Rate die "Genauigkeit" in Termen der Standardabweichung? Können Sie sogar den Vorfaktor der Rate erkennen? Wie müsste er theoretisch heißen?

Lösungsvorschlag:

- `.pdf`-Version: L.6.1
- `.R`-Version: Blatt 6, Aufgabe 1
- `.R`-Version (www): Blatt 6, Aufgabe 1

A.6.2 Visualisierung von Zufallsgrößen — ohne Lösung

(a) Simulieren Sie $n = 500$ Paare von Würfen mit einem gezinkten Würfel mit folgenden Wahrscheinlichkeiten (Stichwort: **sample** und Argument `prob`).

Zahl	1	2	3	4	5	6
W-keit	0.1	0.1	0.1	0.1	0.2	0.4

(b) Bilden Sie jeweils die Produkt P und das Minimum M der beiden Würfe.

(c) Stellen Sie die Paare (P, M) geeignet in einem Plot dar.

A.6.3 Numerische Integration: Berechnung von π

Betrachten Sie den Kreis

$$K = \{(x, y) \in \mathbb{R}^2 \,|\, (x - 0.5)^2 + (y - 0.5)^2 \leq 0.25\}$$

und das Einheitsquadrat

$$E = \{(x, y) \in \mathbb{R}^2 \,|\, 0 \leq x \leq 1; 0 \leq y \leq 1\}$$

(a) Generieren Sie $n = 1000$ Realisationen einer Zufallsvariable U, welche uniform auf $[0, 1] \times [0, 1]$ verteilt ist.

(b) Berechnen Sie das Verhältnis aus Punkten im Kreis zur Gesamtzahl n der gezogen Punkte. Warum schätzt dies das Verhältnis von der Fläche des Kreises zur Fläche des Quadrates? Wie läßt sich damit ein Näherungswert für π angeben?

(c) Wie groß ist n zu wählen, um π mit einer Wahrscheinlichkeit von 95% mit einem Fehler $< 10^{-6}$ zu schätzen.

weitere, nicht stochastische und in dieser Situation wesentlich besser geeignete Verfahren zur Berechnung von π siehe auch http://www.jjj.de/hfloat/hfloatpage.html

Lösungsvorschlag:

- .pdf-Version: L.6.3
- .R-Version: Blatt 6, Aufgabe 3
- .R-Version (www): Blatt 6, Aufgabe 3

A.6.4 Berechnung von $\mathrm{E}[\chi_1^2]$

(a) Sei $Y \sim \chi_1^2$. Berechnen Sie $I = \mathrm{E}[Y]$ analytisch.

(b) Berechnen Sie I in R als Mittelwert von 100 χ_1^2 verteilten Variablen. Schätzen Sie die Varianz Ihrer Näherung, indem Sie diesen Versuch $M = 10000$ mal wiederholen — keine **for**–Schleife, bitte. Können Sie die Varianz analytisch angeben?

(c) Sie kennen die Dichte f_χ von Y — in R ist sie als **dchisq()** implementiert. Wählen Sie geschickt einen Abschneidepunkt T, und berechnen Sie I in R als Mittelwert von 100 Werten $f_\chi(U)$, $U \sim \mathrm{ufo}[0, T]$. Schätzen Sie auch hier die Varianz Ihrer Näherung, indem Sie diesen Versuch $M = 10000$ mal wiederholen.

(d) Verwenden Sie in (c) jeweils auch die Werte $T - U$ als Ausgangswerte ($\rightsquigarrow$ antithetische Variablen). Wie stark reduziert sich die Varianz Ihrer Näherung? Um fair zu bleiben, dürfen Sie nur je 50 Werte U und $T - U$ verwenden.

(e) Sie wissen $\mathcal{L}(Y) = \mathcal{L}(X^2)$, $X \sim \mathcal{N}(0,1)$. Berechnen Sie I in R als Mittelwert von 100 Werten (X^2), $X \sim \mathcal{N}(0,1)$. Können Sie hier einen Unterschied zu dem Ergebnis aus (a) feststellen?

Lösungsvorschlag:

- `.pdf`-Version: L.6.4
- `.R`-Version: Blatt 6, Aufgabe 4
- `.R`-Version (www): Blatt 6, Aufgabe 4

A.6.5 Konfidenzintervalle, Bootstrap

(a) Erzeugen Sie eine Stichprobe von 21 u.i.v. N(0,1) verteilten Zufallsvariablen und berechnen Sie deren (emprischen) Median m_{21}. Geben Sie mit Hilfe der in der Vorlesung in Abschnitt 2.8.3 vorgestellten Formel die asymptotische Verteilung des Medians an, d.h. gehen Sie vor wie in Lösung 1 in Abschnitt 2.8.3. Wie lautet also die Streuung s_{21} von m_{21}, die sich aus der asymptotischen Normalität ergibt?

Bemerkung: Sie erhalten mit $I_{21} := [m_{21} - 1.96 s_{21}; m_{21} + 1.96 s_{21}]$ für $m_{21} \sim \mathcal{N}(0, s_{21}^2)$ ein sogenanntes *95% Konfidenzintervall,* d.h. ein (zufälliges)Intervall I_{21}, das den wahren Median mit 95% Wahrscheinlichkeit überdeckt.

(b) Simulieren Sie nun 10000 Stichproben von 21 u.i.v. N(0,1) verteilten Zufallsvariablen und bestimmen Sie den empirischen Median, die empirische Streuung und geben Sie damit ein (empirisches) 95% Konfidenzintervall für den Median an.

(c) Nehmen Sie die Stichprobe aus Teil (a) und erzeugen Sie hieraus 1000 Bootstrap-Stichproben der Länge 21. Bestimmen Sie dann den gebootstrapten Median, die gebootstrapte Streuung des Median und berechnen Sie damit das zugehörige 95% Bootstrap-Konfidenzintervall für den Median.

Erweiterung: Um die Genauigkeit des Ergebnisses für das Bootstrap zu erhöhen, erzeugen Sie 25 Stichproben von 21 u.i.v. N(0,1) verteilten Zufallsvariablen und führen für jede dieser Stichproben das Bootstrap-Verfahren durch. Mitteln Sie die Ergebnisse für den Median und die Grenzen der Konfidenzintervalle.

(d) Vergleichen Sie die Ergebnisse aus den Teilaufgaben!

Lösungsvorschlag:

- `.pdf`-Version: L.6.5
- `.R`-Version: Blatt 6, Aufgabe 5
- `.R`-Version (www): Blatt 6, Aufgabe 5

A.6.6 Konfidenzintervalle, Bootstrap II —ohne Lösung

(a) Erzeugen Sie eine Stichprobe von 21 u.i.v. N(0,1) verteilten Zufallsvariablen und berechnen Sie deren (empirischen) MAD mad_{21}.
Asymptotisch gilt falls F symmetrisch:

$$\sqrt{n}\,(\mathrm{mad}_n - \mathrm{MAD}(F)) \circ F^n \quad \xrightarrow{\mathrm{w}} \quad \mathcal{N}(0, \frac{1}{16 f(\mathrm{MAD}(F))})$$

also im Falle $F = \mathcal{N}(0,1)$

$$\sqrt{n}\,(\mathrm{mad}_n - \Phi^{-1}(3/4)) \circ \mathcal{N}(0,1)^n \quad \xrightarrow{\mathrm{w}} \quad \mathcal{N}(0, 0.6182)$$

Wie lautet also die aus der asymptotischen Varianz zu gewinnende Streuung s_{21} von mad_{21}?

Bemerkung: Sie erhalten mit $I_{21} := [\mathrm{mad}_{21} - 1.96 s_{21}; \mathrm{mad}_{21} + 1.96 s_{21}]$,

Achtung wir sind wirklich am MAD interessiert und nicht an einer Schätzung für σ, d.h. Sie müssen Argument `constant=1` in **mad** übergeben!

unterstellt, dass $\mathrm{mad}_{21} \sim \mathcal{N}(0, s_{21}^2)$, ein sogenanntes *95% Konfidenzintervall,* d.h. ein (zufälliges)Intervall I_{21}, das den wahren MAD mit 95% Wahrscheinlichkeit überdeckt.

(b) Simulieren Sie nun 10000 Stichproben von 21 u.i.v. N(0,1) verteilten Zufallsvariablen und bestimmen Sie den empirischen MAD, dessen empirische Streuung und geben Sie damit ein (empirisches) 95% Konfidenzintervall für den Median an.

(c) Nehmen Sie die Stichprobe aus Teil (a) und erzeugen Sie hieraus 1000 Bootstrap-Stichproben der Länge 21. Bestimmen Sie dann den gebootstrapten MAD, die gebootstrapte Streuung des MAD und berechnen Sie damit das zugehörige 95% Bootstrap-Konfidenzintervall für den MAD.

(d) Vergleichen Sie die Ergebnisse aus den Teilaufgaben!

A.7 Blatt 7

- Stoff bis einschließlich Abschnitt 3.6 — Rekursionen und Frames
- Abschnitt 3.4 (Debugging) wird in Aufgabe A.9.2 aufgegriffen

A.7.1 Maximale Lücke

Motivation: Die in der Numerik vorgestellten, deterministischen Methoden, um Integrale der Form $I(f) := \int_a^b f(x)\,dx$ numerisch zu berechnen, sind für niedrige Dimensionen der stochastischen *Monte Carlo Integration* weit überlegen. Grund dafür ist die Eigenschaft stochastisch gezogener Stützstellen, sich in niedrigen Dimensionen zu "verklumpen". Diesen Effekt mathematisch in den Griff zu bekommen ist nicht trivial; auf Simulationsbasis lässt er sich aber recht einfach untersuchen: Man betrachtet dabei die Verteilung der größten Lücke einer Stichprobe $X_1, \ldots, X_n$ — hier im zweidimensionalen: $L_n := \min_{i \neq j} \|X_i - X_j\|$ mit $\|\cdot\|$ dem Euklidischen Abstand im $\mathbb{R}^2$.

Generieren Sie $M = 500$ Stichproben $X_1, \ldots, X_n$ der Länge n, wobei X_i die Realisationen einer 2-dimensionalen Zufallsvariable X seien, welche uniform auf $[0,1]^2$ verteilt ist, d.h. $\mathcal{L}(X) = \mathrm{Ufo}\left([0,1]^2\right)$. Betrachten Sie die Fälle $n = 5, 25, 100$ und 500. Berechnen Sie dann für jede der Stichproben $\min_{i \neq j} \left(\|X_i - X_j\|\right)$, wobei $\|\cdot\|$ der Euklidische Abstand im $\mathbb{R}^2$ ist. Betrachten Sie das "Summary" der

Minima und stellen Sie die Minima mit Hilfe von Boxplots und Histogrammen graphisch dar. Interpretieren Sie die Ergebnisse!

Lösungsvorschlag:

- `.pdf`-Version: L.7.1
- `.R`-Version: Blatt 7, Aufgabe 1
- `.R`-Version (www): Blatt 7, Aufgabe 1

A.7.2 Buffons Nadelproblem — Berechnung von π II

(a) Geben Sie dazu $d = 2$ im Quadrat $Q = [-10, 10]^2$ vor — erste Streifengrenze bei $x = -10$ — und ziehen Sie in Q uniform den Mittelpunkt einer jeden Nadel. Der Winkel der Nadel zur x–Achse sei ebenfalls gleichverteilt auf $[0, 2\pi]$. Legen Sie die x– und y–Koordinaten der beiden Endpunkte der Nadeln in Variablen `xl`, `xr`, `yl`, `yr` ab.

(b) Finden Sie heraus, wie man analytisch anhand der Werte `xl`, `xr`, `yl`, `yr` und der Koordinaten der Streifengrenzen bestimmen kann, ob es zum Schnitt einer Nadel mit einer Streifengrenze kommt.

Hinweis: Um **for**–Schleifen zu vermeiden bilden Sie eine Matrix M in Dimensionen $N \times m$, N die Zahl der Nadeln und m die Zahl der Streifengrenzen, deren Eintrag $M_{i,j}$ gerade die x–Koordinate des j–ten Streifengrenzes ist. Betrachten Sie die Vorzeichen von $\texttt{xl} - M$ und $\texttt{xr} - M$.

(c) Berechnen Sie einen Indikator (Vektor mit Werten $0, 1$ oder `T`, `F`), der anzeigt, ob Nadel i einen Schnitt mit den Streifengrenzen hat oder nicht.

(d) Berechnen Sie hieraus eine Schätzung $\hat{\pi}$ für π; wie ist $N/\hat{\pi}$ verteilt? Berechnen Sie in R zu vorgegebenen d_l, d_r, $-\pi < d_l < 0 < d_r$, die Wahrscheinlichkeit $\mathbb{P}(d_l < \hat{\pi} - \pi \leq d_r)$. Finden Sie das minimale N, so dass mit $-d_l = d_r = 10^{-3}$ diese Wahrscheinlichkeit größer ist als 95%.

(e) Plotten Sie die Nadeln und die Streifen. Verwenden Sie dabei die Datei `buffon.r` auf der Service–Page als Hilfe–Stellung / Anregung.

(f) Vergleichen Sie die Genauigkeit mit der der Methode aus Aufgabe A.6.3.

Lösungsvorschlag:

- `.pdf`-Version: L.7.2
- `.R`-Version: Blatt 7, Aufgabe 2
- `.R`-Version (www): Blatt 7, Aufgabe 2

A.7.3 Dichteplot

(a) Schreiben Sie eine Funktion, welche als Übergabeparameter den Namen einer beliebigen Verteilungsklasse (z.B. norm, chisq, ...) und die für die Verteilungsklasse benötigten Parameter (z.B. **mean**, **sd**, **df**, ...) erhält. Zusätzlich werde eine Variable übergeben, die angibt, ob die Verteilung diskret oder stetig ist. Die Funktion gebe als Ergebnis das 1-te, 2-te (Median) und 3-te Quartil der jeweiligen Verteilung zurück und plote die Dichte (stetige Verteilung) bzw. Wahrscheinlichkeitsfunktion (diskrete Verteilung) im Bereich vom 5% bis zum 95%-Quantil. Dabei werde für stetige Verteilungen ein Linienplot und für diskrete Verteilungen ein "step-plot" erzeugt.

Hinweis: Auf der Service–Seite findet sich in der Datei `dichte.r` Funktion dichte(), die ähnliches leistet.

(b) Erzeugen Sie mit Hilfe der Funktion **prompt()** eine Hilfedatei zu Ihrer Funktion aus Teil (a).

Lösungsvorschlag:

- `.pdf`-Version: L.7.3
- `.R`-Version: Blatt 7, Aufgabe 3
- `.R`-Version (www): Blatt 7, Aufgabe 3

A.7.4 Aufzählung mehrerer Alternativen mit if und switch

Betrachten Sie die drei Anweisungen

```
plot(qnorm, from=0, to=1)
plot(pnorm, from=-5, to=5)
plot(dnorm, from=-5, to=5)
```

Schreiben Sie eine Funktion `myplot()` mit Argument `was` vom Typ **character**. `was` kann die Werte `"Quantilsfunktion"`, `"Verteilungsfunktion"` oder `"Dichtefunktion"` annehmen. Versehen Sie dabei Argument `was` mit einem geeigneten Defaultwert. Anhand von `was` soll entschieden werden, welche der drei Anweisungen ausgeführt werden soll. Tun Sie dies

(a) mit **if** - Verzweigungen

(b) mit **switch** - Verzweigungen

(c) mit **switch** - Verzweigungen und **pmatch**

A.7.5 Schleifen – Schleifenvermeidung – Laufzeitvergleich

(a) Gegeben seien zwei Vektoren `a` und `b`, wobei `a` sehr viel größere Dimension besitze als `b`. Berechnen Sie hieraus einen Vektor `s`, der als Einträge `s[i]` die Anzahl der Einträge von `a` mit `a[j]<= b[i]` enthält. Verwenden Sie hierzu:

- **for**-Schleifen
- **while**-Schleifen
- **repeat**-Schleifen
- keine Schleifen

(b) Schreiben Sie um Ihre Programme aus Teil (a) herum eine Funktion `vergleich()`, welche als Übergabe-Parameter a, b und das zu verwendende Verfahren habe.

(c) Erzeugen Sie nun ein Paar a, b, wobei a ein Vektor von 5000 Realisationen einer poisson-verteilten Zufallsvariable mit

Parameter **lambda=8** und b ein Vektor von 50 Realisationen einer poisson-verteilten Zufallsvariable mit Parameter **lambda=12** sei. Lassen Sie s mit jedem der Verfahren berechnen und vergleichen Sie dabei mit Hilfe der Funktion **system.time()** die jeweiligen Rechenzeiten.

Lösungsvorschlag:

- `.pdf`-Version: L.7.5
- `.R`-Version: Blatt 7, Aufgabe 5
- `.R`-Version (www): Blatt 7, Aufgabe 5

A.7.6 Adaptives Verfahren zur zweidimensionalen numerischen Integration

Gegeben sei eine Funktion $f\colon \mathbb{R}^2 \to \mathbb{R}$. Es soll eine numerische Näherung für $I_2(f,a,b,c,d) := \int_a^b \int_c^d f(x,y)\,dx\,dy$ berechnet werden.

(a) Schreiben Sie eine Funktion, die dies adaptive (per Rekursion) erledigt. Gehen Sie dabei wie folgt vor:

(1) Ziehen Sie $M = 10$ Punkte aus $\mathrm{Ufo}([a,b] \times [c,d])$ und werten Sie die Funktion f an diesen Punkten aus. Wiederholen Sie dies 2 Mal, bilden Sie jeweils die Mittelwerte der Funktionswerte und multiplizieren diese mit der Fläche des Rechtecks, d.h. mit $(b-a)(d-c)$.

(2) Ist die Abweichung größer als der vorgegebende Fehler, so splitten Sie $[a,b]$ auf in $[a,m_1]$ und $[m_1,b]$ mit $m_1 = \frac{a+b}{2}$ und wiederholen Schritt (1) für die beiden "Teil-Vierecke" (ohne jedoch den Fehler zu halbieren)

(3) Ist die Abweichung in einem der beiden neuen "Teil-Vierecke" (bzw. in beiden) erneut größer als der vorgegebene Fehler, so splitten Sie dieses Mal in y-Richtung auf, d.h. $[c, d]$ wird ersetzt durch $[c, m_2]$ und $[m_2, d]$ mit $m_2 = \frac{c+d}{2}$ und wiederholen Schritt (1) für die entstandenen "Teil-Vierecke" (ohne jedoch den Fehler zu halbieren).

(4) Schreiben Sie eine Funktion, die (1)-(3) durchführt und rufen Sie die Funktion mit `recall ()` rekursiv auf bis für jedes der "Teil-Vierecke" der vorgegebene Fehler unterschritten wird bzw. die maximale Rekursionstiefe erreicht ist. Führen Sie zusätzlich einen Zähler mit, der Ihnen die Anzahl der Verdoppelungen protokolliert und der Ihnen angibt, ob die x-Seiten (Zähler ungerade) oder die y-Seiten (Zähler gerade) der Vierecke zu halbieren sind.

Hinweis: Orientieren Sie sich an der Funktion `area()` aus Beispiel 3.6-2 der Vorlesung.

(b) Betrachten Sie für $z := (x, y)'$ die β-Funktion von $|z| = \sqrt{x^2 + y^2}$, d.h. die Funktion

$$f_{\alpha,\beta}(z) = |z|^{\alpha-1} \cdot (1 - |z|)^{\beta-1} \cdot \mathrm{I}(|z| \leq 1)$$

Berechnen Sie das Integral $I_2(f_{3,2}, -1, 1, -1, 1)$ mit Hilfe Ihrer Funktion aus Teil (a) und protokollieren Sie die Auswertungsstellen. Geben Sie hierzu 10 als maximale Rekursionstiefe und 10^{-5} als Fehler vor.

Hinweis: Ziehen Sie hierzu Beispiel 3.6-5 aus der Vorlesung heran. Zum Vergleich: Der exakte Wert dieses Integrals ist: $\frac{\pi}{10}$.

(c) Plotten Sie die Auswertungsstellen.

Lösungsvorschlag:

- `.pdf`-Version: L.7.6
- `.R`-Version: Blatt 7, Aufgabe 6
- `.R`-Version (www): Blatt 7, Aufgabe 6

A.8 Blatt 8

- Stoff bis einschließlich Abschnitt 4.7.2 — Export von Graphik

Peter Ruckdeschel
Matthias Kohl

A.8.1 Visualisierung

(a) Visualisieren Sie Aufgabe A.6.3 — plotten Sie Kreis und Quadrat und färben Sie die Punkte je nachdem, ob sie im Kreis liegen oder nicht in rot oder grün.

(b) Visualisieren Sie eine Stichprobe aus Aufgabe A.7.2 — zeichnen sie die minimale Lücke als grüne Linie ein.

(c) Plotten Sie $M = 100$ Realisationen (Pfade) eines zweidimensionalen Random Walks der Länge $T = 100$, d.h. M Folgen der Zufallsvariablen $X_t = (X_{t,1}, X_{t,2})^\tau$, $t = 1, \ldots T$, $X_t = X_{t-1} + U_t$, $U_t = (U_{t,1}, U_{t,2})^\tau$, $U_{t,i} \overset{\text{u.i.v.}}{\sim} \pm 1$, $i = 1, 2$ jeweils mit Wahrscheinlichkeit $1/2$. Färben Sie die Pfade rot ein, bei denen mindestens 4 aufeinanderfolgende Zeitpunkte in einem Quadranten liegen.

Lösungsvorschlag:

- `.pdf`-Version: L.8.1
- `.R`-Version: Blatt 8, Aufgabe 1
- `.R`-Version (www): Blatt 8, Aufgabe 1

A.8.2 Bundestagswahl 2005

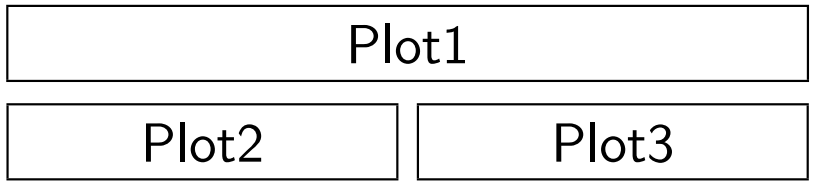
(a) Beziehen Sie von der Homepage des statistischen Bundesamts die Wahlergebnisse (absolute Stimmenzahlen) der Bundestagswahl 2002 und teilen Sie die Ergebnisse auf in CDU/CSU, SPD, Grüne, FDP, PDS, "Sonstige" und Nichtwähler. Beschränken Sie sich dabei auf 5 Bundesländer Ihrer Wahl. Erzeugen Sie einen `Dataframe`, wobei Sie die Spalten den Bundesländern und die Zeilen den verschiedenen Gruppen (Parteien) zuordnen.

Link: http://www.bundeswahlleiter.de

(b) Generieren Sie mit Hilfe von **matplot()** und **barplot()** graphische Darstellungen der Ergebnisse. Im Fall von **matplot()** soll jede Kurve dem Wahlergebnis einer Gruppe (Partei) entsprechen, wobei Sie für jede Gruppe (Partei) eine andere Farbe verwenden. Im Fall von **barplot()** soll jede Säule einem Bundesland

entsprechen, wobei Sie die Gruppen (Parteien) einmal mit Schraffierungen und einmal mit Farben unterscheiden.

(c) Stellen Sie die drei Plots zusammen auf einer Seite dar von der Form:



Hinweis: Gehen Sie vor wie in Abschnitt 4.2.2

Lösungsvorschlag:

- `.pdf`-Version: L.8.2
- `.R`-Version: Blatt 8, Aufgabe 2
- `.R`-Version (www): Blatt 8, Aufgabe 2

A.8.3 Multivariate Konvexkombination

(a) Erzeugen Sie $n = 100$ Realisationen von Paaren $Z = (Z_1, Z_2)$ unabhängig identisch verteilter Normalverteilungen $\mathcal{N}(0, 1)$. Erzeugen Sie für jede Realisation hieraus Realisationen der Zufallsvariablen X, welche die folgende Verteilung besitzt:

$$\mathcal{L}(X) = 0.9 \cdot \mathcal{L}(X_{\text{id}}) + 0.1 \cdot \mathcal{L}(X_{\text{cont}})$$

mit $\mathcal{L}(X_{\text{id}}) = \mathcal{L}(\mu_{\text{id}} + S_{\text{id}} Z)$ und $\mathcal{L}(X_{\text{cont}}) = \mathcal{L}(\mu_{\text{cont}} + S_{\text{cont}} Z)$, wobei $\mu_{\text{id}} = (0, 0)'$, $\mu_{\text{cont}} = (1, 1)'$,

$$S_{\text{id}} = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}, \qquad S_{\text{cont}} = \begin{pmatrix} 0.64 & 0 \\ 0 & 0.64 \end{pmatrix}$$

Protokollieren Sie dabei mit, welche Realisationen aus der kontaminierenden Verteilung $\mathcal{L}(X_{\text{cont}})$ stammen.

Hinweis: Vergleiche Aufgabe A.5.1 bzw. das entsprechende R-File.

(b) Erzeugen Sie eine ellipsenförmige $(1-\alpha)$-Konfidenzregion für die ideale Verteilung $\mathcal{L}(X_{\text{id}})$. Gehen Sie wie folgt vor:

- Berechnen Sie die Singulärwertzerlegung (**svd()**) von S^2_{id} in die Matrizen d, u, v, wobei d eine Diagonalmatrix ist und in d nur die Diagonalelemente abgelegt sind.
- Bestimmen Sie dann für ein θ-Gitter, $\theta \in [0, 2\pi]$, die Werte von $x := r\cos\theta\sqrt{d_1}$ und $y := r\sin\theta\sqrt{d_2}$, wobei $r :=$ **qnorm**$(1-\alpha/2)$ und d_1, d_2 die Diagonalelemente von d sind. Wählen Sie $\alpha = 0.05$.
- Multiplizieren Sie (Matrixmultiplikation) die Matrix u mit der Matrix, die als Zeilen x und y enthält und plotten Sie die Zeilen der Ergebnismatrix gegeneinander.

(c) Ergänzen Sie den Plot nun um die in Teil (a) generierten Daten. Dabei zeichnen Sie die Punkte aus der kontaminierenden Verteilung rot, falls sie innerhalb der Ellipse liegen und orange, falls sie außerhalb liegen. Die Punkte aus der idealen Verteilung

zeichnen Sie grün, falls Sie innerhalb der Ellipse liegen und blau, falls sie außerhalb liegen. Für die Fallunterscheidungen multiplizieren Sie Ihre Beobachtungen X mit u, d.h. $Y = (Y_1, Y_2)' = u\%*\%X$ und kontrollieren, ob

$$\frac{Y_1^2}{d_1} + \frac{Y_2^2}{d_2} \leq r^2$$

Lösungsvorschlag:

- `.pdf`-Version: L.8.3
- `.R`-Version: Blatt 8, Aufgabe 3
- `.R`-Version (www): Blatt 8, Aufgabe 3

A.8.4 Regressionsplots

(a) Erzeugen Sie $n = 16$ Realisationen einer Zufallsvariable ϵ mit Verteilung $\mathcal{N}(0, 1)$ und addieren Sie zu dieser das $\theta = 2$-fache des Vektor $X = (-5, -4, \ldots, 0, 1, \ldots, 10)$ und die Konstante $\alpha = 1$, d.h. $Y := \theta X + \alpha + \epsilon$.

(b) Wenden Sie auf Y aus Teil (a) die Exponentialfunktion **exp()** an und plotten Sie die Punkte $Z(X) := \exp Y(X)$. Fügen Sie dem Plot die Kurve $Z = \exp(\theta X + \alpha)$ hinzu.

(c) Erzeugen Sie einen zweiten Plot von $Z(X)$, wobei Sie dieses Mal die Z-Achse logarithmisch transformieren. Beschriften Sie die Z-Achse sowohl mit den Werten von $\log(Z) = Y$ als auch mit den Werten von Z. Ergänzen Sie abschließend den Plot noch um die Kurve $Y = \theta X + \alpha$.

Lösungsvorschlag:

- `.pdf`-Version: L.8.4
- `.R`-Version: Blatt 8, Aufgabe 4
- `.R`-Version (www): Blatt 8, Aufgabe 4

A.8.5 Regressionsplots II —ohne Lösung

(a) Erzeugen Sie $n = 16$ Realisationen einer Zufallsvariable ϵ mit Verteilung $\mathcal{N}(0, 1)$ und addieren Sie zu dieser das $\theta = 2$-fache des Vektor $X = (-5, -4, \ldots, 0, 1, \ldots, 10)$ und das $\rho = -2$-fache des Vektors $U = (|X| + 1)^{-1}$ sowie die Konstante $\alpha = 1$, d.h. $Y := \theta X + \rho U + \alpha + \epsilon$.

(b) Wenden Sie auf Y aus Teil (a) die logistische Funktion $l(x) = (1 + e^{-x})^{-1}$ an und plotten Sie die Punkte $Z(X) := l(Y(X))$. Fügen Sie dem Plot die Kurve $Z = l(\theta X + \rho U + \alpha)$ hinzu.

(c) Erzeugen Sie einen zweiten Plot von $Z(X)$, wobei Sie dieses Mal die Z-Achse mit der Transformation $j(x) = \log(x/(1 - x))$ transformieren. Beschriften Sie die Z-Achse sowohl mit den Werten von $j(Z) = Y$ als auch mit den Werten von Z. Ergänzen Sie abschließend den Plot noch um die Kurve $Y = \theta X + \rho U + \alpha$.

(d) Verwenden Sie bei der Beschriftung die "richtigen" griechischen Buchstaben (siehe ?plotmath)...

A.9 Blatt 9

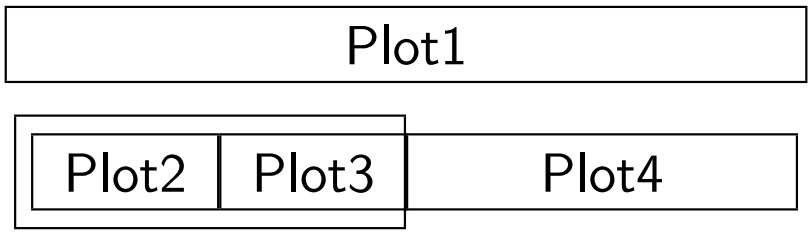
- Stoff bis einschließlich Abschnitt 4.7.2 — Export von Graphik

A.9.1 3D-Plot

Sei K der Ausübungspreis einer Europäischen Call-Option auf eine Aktie X zum Zeitpunkt T. Dann ist zum Zeitpunkt $t < T$ bei konstanter Volatilität σ konstantem Zins r und Aktienkurs X_t für $h = \frac{-r(T-t)+\log K/X_t}{\sqrt{T-t}\,\sigma}$, sowie Φ die Verteilungsfunktion der Standardnormalverteilung der Black-Scholes-Preis

$$c = X_t\Phi(-h+\sigma\sqrt{T-t}\,/2) - Ke^{-r(T-t)}\Phi(-h-\sigma\sqrt{T-t}\,/2)$$

Plotten Sie diesen Preis für $K = 10$, $X_t = 12$, $T = 0.8$ (Jahre), $r = 5\%$ und $\sigma \in [0.05, 0.6]$ sowie $t \in [0, 0.8]$ — einmal als 3D-plot (**image**), als 3D-plot (**persp**), einmal als Niveauplot, einmal mit "vielen Linienmit **matplot** Stellen Sie die vier Plots zusammen auf einer Seite dar von der Form:



Lösungsvorschlag:

- `.pdf`-Version: L.9.1
- `.R`-Version: Blatt 9, Aufgabe 1
- `.R`-Version (www): Blatt 9, Aufgabe 1

A.9.2 Powerpoint-Präsentation

Bereiten Sie einen der Plots aus den Aufgaben A.8.1-A.8.4 für eine Powerpoint-Präsentation vor. Ergänzen Sie den Plot vorher, d.h. falls noch nicht geschehen, um eine passende Überschrift, Untertitel und Legende.

Hinweis: Gehen Sie vor wie in Abschnitt 4.7.2

Lösungsvorschlag:

- `.pdf`-Version: L.9.2

A.9.3 Farben

Definieren Sie mit dem Paket `RColorBrewer` eine farbenblindengerechte Beamerpalette mit 7 divergierenden Farben.

Lösungsvorschlag:

- `.pdf`-Version: L.9.3
- `.R`-Version: Blatt 9, Aufgabe 3
- `.R`-Version (www): Blatt 9, Aufgabe 3

A.9.4 Umgang mit GGobi/Gtk+ mit dem Plato-Datensatz —ohne Lösung

(a) Laden Sie sich unter http://www.ggobi.org/ das Programm ggobi sowie unter http://www.gtk.org/ das Toolkit für plattformunabhängige Benutzeroberflächen GTK+ herunter und installieren Sie es sich wie dort beschrieben.

(b) Ebenso laden Sie sich die R-Pakete rggobi und RGtk2 herunter.

(c) Importieren Sie in ggobi den Datensatz plato.xml von der Serviceseite und lesen Sie sich die entsprechende Seite plato.html durch. Versuchen Sie anhand von ggobi eine chronologische Anordnung der Werke Platos vorzunehmen.

A.9.5 Chernoff-Gesichter

(a) Laden Sie sich die Pakete `aplpack` und `alr3` herunter und binden Sie diese in Ihrer R-Sitzung mit **require** ein.

(b) Holen Sie sich aus `alr3` den Datensatz `banknote` in Ihren R-Workspace; informieren Sie sich über diesen Datensatz (schreiben Sie kurz, was Sie herausgefunden haben in einen Kommentar).

(c) Erstellen Sie für diesen für die Scheine 1–10 und 101–110 *Chernoff Gesichter* mit dem Befehl `faces`. Ziehen Sie jetzt 30 Scheine aus den 200 (ohne Zurücklegen) und versuchen Sie, zu entscheiden, welche Scheine echt sind.

Lösungsvorschlag:

- `.pdf`-Version: L.9.5
- `.R`-Version: Blatt 9, Aufgabe 5
- `.R`-Version (www): Blatt 9, Aufgabe 5

A.9.6 R und TclTk

Schauen Sie sich den Source-Code zum Demo `tkdensity` (nach Einladen mit **require** verwenden Sie dazu **demo(density)**) im Paket `tcltk` an. Wandeln Sie dieses soweit ab, dass Sie in Aufgabe A.9.1 für den Plot $t \mapsto c(t, T, \sigma, r, K, X_t)$ sämtliche weiteren Eingabeparameter, also T, σ, r, K, X_t über Slider steuern können. Wenn Sie wollen, können Sie auch in einem 3D-plot den Perspektiv-Winkel über einen Slider steuern.

Lösungsvorschlag:

- `.pdf`-Version: L.9.6
- `.R`-Version: Blatt 9, Aufgabe 6
- `.R`-Version (www): Blatt 9, Aufgabe 6

A.10 Blatt 10

- Stoff bis einschließlich Abschnitt 5.1 — Testtheorie

A.10.1 Shapiro-Wilk, Kolmogorov-Smirnov, χ^2-Anpassungstest

Testen Sie mit Hilfe von *Shapiro-Wilk*, *Kolmogorov–(Smirnov)* und χ^2*-Anpassungs-Test*, ob die Nullhypothese, dass die Daten im `normal` Datensatz auf der Service–Homepage $\mathcal{N}(1, 3)$-verteilt sind, bei einem Signifikanzniveau von 10% abgelehnt werden muss. Erstellen Sie zusätzlich einen qq–Plot der Daten.

Lösungsvorschlag:

- `.pdf`-Version: L.10.1
- `.R`-Version: Blatt 10, Aufgabe 1
- `.R`-Version (www): Blatt 10, Aufgabe 1

A.10.2 Wilcoxon und t-Test, χ^2- und F-Test

(a) Auf der Service–Homepage finden Sie den Datensatz `uscomp`. Testen Sie mit Hilfe des *Wilcoxon* und des *t-Tests*, ob die Nullhypothese, dass die Mittelwerte der Sektoren *Energie* und *Finanzen* für die Variable *X6* gleich groß sind, bei einem Signifikanzniveau von 10% abgelehnt werden muss. Wie groß sind die jeweiligen *p-Werte*?

(b) Testen Sie mit Hilfe des χ^2-Tests, ob die Nullhypothese, dass die Varianz des Sektors *Energie* für die Variable *X6* den Wert 100 hat, bei einem Signifikanzniveau von 10% abgelehnt werden muss.

(c) Testen Sie mit Hilfe des *F-Tests*, ob die Nullhypothese, dass die Varianzen der Sektoren *Energie* und *Finanzen* für die Variable *X6* gleich groß sind, bei einem Signifikanzniveau von 10% abgelehnt werden muss. Welchen *p-Wert* erhalten Sie?

Lösungsvorschlag:

- `.pdf`-Version: L.10.2
- `.R`-Version: Blatt 10, Aufgabe 2
- `.R`-Version (www): Blatt 10, Aufgabe 2

A.10.3 Fisher- und t-Test

(a) Ermitteln Sie im `kredit1` Datensatz aus Aufgabe A.4.3 den Anteil der Arbeitslosen in der Gruppe der "Kreditwürdigen" bzw. der "Kreditunwürdigen".

(b) Testen Sie mit Hilfe des *exakten Fishertests*, der asymptotischen Variante des *exakten Fishertests* und des *t-Tests*, ob die Nullhypothese, dass der Anteil in beiden Gruppen gleich gross ist, bei einem Signifikanzniveau von 10% abgelehnt werden muss. Wie gross sind die *p-Werte* bei den letzten beiden Tests?

Lösungsvorschlag:

- `.pdf`-Version: L.10.3
- `.R`-Version: Blatt 10, Aufgabe 3
- `.R`-Version (www): Blatt 10, Aufgabe 3

A.10.4 Testvergleich durch Simulation

Seien $X_1, \ldots, X_n \overset{\text{u.i.v.}}{\sim} \mathcal{N}(\theta, 1)$. Wir wollen testen, ob $H_0: \theta = 0$ oder $H_1: \theta = 1$ zutrifft. Als Test verwenden wir einmal den Neyman–Pearson–Test (NPT) und andererseits einen Test, basierend auf der Zahl N der Beobachtungen, so dass $X_i > 0$ ist — ein Vorzeichentest (VZT) also.

(a) **(freiwillig)** Bestimmen Sie den NPT zum Niveau $\alpha = 0.05$.

Lösung (NPT): $\phi_0(X_1, \ldots, X_n) = \mathrm{I}_{\{\bar{X}_n \geq q_0\}}$
mit $q_0 =$ **qnorm**(0.95,sd=1/sqrt(10))

Geben Sie die Verteilung von N, der Zahl der positiven Beobachtungen, unter H_0 und unter H_1 an. Bestimmen Sie einen VZT (mit und ohne Randomisierung) zum Niveau $\alpha = 5\%$ für $n = 10$ Beobachtungen zum Niveau $\alpha = 0.05$.

Lösungen (VZT):
unrandomisiert: $\phi_N(N) = \mathrm{I}_{\{N>8\}}$
randomisiert: $\phi_N^*(N) = \mathrm{I}_{\{N>8\}} + \mathrm{I}_{\{N=8\}}\, \mathrm{I}_{\{R=1\}}$, wobei
$R \sim \mathrm{Bin}(1, r_0)$ unabhängig von N mit
$r_0 =$ (**pbinom**(8,10,0.5) $- 0.95$)/**dbinom**(8,10,0.5)

Berechnen Sie die Macht des VZT und vergleichen Sie diese mit der des NPT. Sie haben die (unabh.) Beobachtungen

$$0.1, 2.1, -1.2, 1.0, 1.6, 1.3, 0.2, 0.1, 0.3, 2.2$$

gemacht. Treffen Sie eine Testentscheidung mit dem NPT und dem VZT zum Niveau 5% und geben Sie dazu auch den p–Wert an. Nun ersetzen Sie die erste Beobachtung durch -20.2. Ändert sich Ihre Entscheidung? Welcher der beiden Tests reagiert sensibler auf diesen "Ausreißer". Argumentieren Sie, warum dem so ist.

(b) Simulieren Sie $M = 1000$ mal Stichproben der Länge 10 jeweils unter H_0 unter H_1.

(c) Treffen Sie für jede Stichprobe die Testentscheidung mit dem VZT und dem NPT und berechnen Sie jeweils (empirisch) den Fehler 1. und 2. Art.

(d) Nun simulieren Sie wie folgt Ausreißer–behaftete Stichproben: Zu jeder Variablen $X_{i,j,k}$, $i=1,\ldots,10$, $j=1,\ldots,M$, $k=0,1$ ($\hat{=} H_0$ resp. H_1) erzeugen Sie die Variable $Y_{i,j,k} := (1 - U_{i,j,k})X_{i,j,k} + (-1)^k \cdot 10\, U_{i,j,k}$ mit $U_{i,j,k} \overset{\text{u.i.v.}}{\sim} \text{Bin}(1, 0.05)$ und unabhängig zu den $X_{i,j,k}$.

(eingerahmt in der freiwilligen Teilaufgabe)

(e) Führen Sie wieder für jede der M Stichproben den VZT und den NPT durch und berechnen Sie jeweils (empirisch) den Fehler 1. und 2. Art. Halten die Tests auch unter den kontaminierten Stichproben das Niveau ein? Was passiert mit der Macht?

Lösungsvorschlag:

- `.pdf`-Version: L.10.4
- `.R`-Version: Blatt 10, Aufgabe 4
- `.R`-Version (www): Blatt 10, Aufgabe 4

A.11 Blatt 11

- Stoff bis einschließlich Abschnitt 5.2.3
- ML–Schätzung mit numerischen Methoden wird noch einmal in Aufgabe A.12.2 aufgegriffen

A.11.1 Indiskrete Umfrage

In einer Umfrage soll das Treueverhalten von Ehegatten erfragt werden. Damit weder der Interviewer noch sonst jemand identifizieren kann, welche Antwort der Befragte gegeben hat, geht man so vor: Der Interviewer lässt den Befragten zweimal eine Münze werfen, wobei der Interviewer die Resultate M_1 und M_2 nicht sieht. Ist $M_1 =$ "Kopf", so beantwortet der Befragte die indiskrete Frage, ist $M_1 =$ "Zahl", so gibt er das Resultat des zweiten Münzwurfs an — $(M_2 =$ "Kopf"$)\hat{=}$"ja", $(M_2 =$ "Zahl"$)\hat{=}$"nein". Insgesamt werden $n = 2300$ Männer befragt. Es antworten $S = 682$ mit "ja" auf die Frage "Sind sie untreu?". Geben Sie eine ML–Schätzung $\hat{p}$ für den Anteil der untreuen Männer, sowie ein (auf der Normalapproximation beruhendes) 97.56%–Konfidenzintervall dafür an.

Lösungsvorschlag:

- `.pdf`-Version: L.11.1
- `.R`-Version: Blatt 11, Aufgabe 1
- `.R`-Version (www): Blatt 11, Aufgabe 1

A.11.2 ML–Schätzer für K aus $\mathrm{HypGeo}(N, K, n)$

Berechnen Sie den ML–Schätzer für den Parameter K einer Hypergeometrischen Verteilung $\mathrm{HypGeo}(N, K, n)$. Bei $m = 5$ unabhängigen Ziehungen aus $\mathrm{HypGeo}(N, K, n)$ mit $N = 30$ und $n = 10$ wurden Realisationen $k = 3, 4, 6, 4, 7$ gezogen, d.h. nach jeder Ziehung von $n = 10$ Kugeln wird k ermittelt, und anschließend werden die n Kugeln wieder in die Urne gegeben. Bestimmen Sie den ML–Schätzer $K^{\mathrm{ML}} \in \mathbb{N}_0$, indem Sie die Likelihood numerisch optimieren.

N Gesamtzahl der Kugeln in der Urne, K Zahl der weißen Kugeln in der Urne, n Zahl der Ziehungen ohne Zurücklegen, und die Zufallsgröße k ist die Zahl der gezogenen weißen Kugeln

Lösungsvorschlag:

- `.pdf`-Version: L.11.2
- `.R`-Version: Blatt 11, Aufgabe 2
- `.R`-Version (www): Blatt 11, Aufgabe 2

A.11.3 Simulationsstudie

(a) Schreiben Sie mit Hilfe der Funktion **uniroot()** eine Funktion, die aus einem Vektor `X` beliebiger Länge den M–Schätzer $S^{(\psi)}$ zur ψ–Funktion $\psi(x) = \max(-0.7, \min(x, 0.7))$ ermittelt.

(b) Simulieren Sie $M = 10000$ ideal verteilte Stichproben $X_{i,j}^{\text{id}} \overset{\text{u.i.v.}}{\sim} \mathcal{N}(0,1)$ der Längen $n = 5, 10, 100$ — $i = 1, \ldots, M$, $j = 1, \ldots, n$, sowie entsprechend kontaminiert verteilte Stichproben $X_{i,j}^{\text{cont}} \overset{\text{u.i.v.}}{\sim} 0.9\mathcal{N}(0,1) + 0.1\text{Cauchy}$

(c) Berechnen Sie für jede der M idealen und kontaminierten Stichproben das arithmetische Mittel $(\bar{X}_n)_i^{\text{id/cont}}$, den Median $(\text{Med}_n)_i^{\text{id/cont}}$ und den M–Schätzer $(\text{S}_n^{(\psi)})_i^{\text{id/cont}}$.

(d) Fertigen Sie jeweils für n fixiert gemeinsame Boxplots der drei Schätzer jeweils in der idealen und in der kontaminierten Situation an, sowie zur besseren Unterscheidung bei Kontamination einen mit Median und M–Schätzer. Insgesamt also (3×3) Plots in ein Graphikfenster.

(e) Schätzen Sie die Varianzen und mittleren quadratischen Fehler (MSE) der Schätzer in der idealen und in der kontaminierten Situation durch die Stichprobenvarianz bzw. den Stichproben–MSE.

Lösungsvorschlag:

- `.pdf`-Version: L.11.3
- `.R`-Version: Blatt 11, Aufgabe 3
- `.R`-Version (www): Blatt 11, Aufgabe 3

A.12 Blatt 12

- Stoff bis einschließlich Abschnitt 6.6 — sich selbst verändernde Programme

A.12.1 Berechnung eines Quantils

Gegeben sei die Funktion $F(x) :=$**pnorm**(x) (die Verteilungsfunktion der Standard-Normalverteilung $\mathcal{N}(0,1)$). Berechnen Sie x_0 so, dass $F(x_0) - 0.95 = 0$, d.h. x_0 ist das 95%-Quantil von $\mathcal{N}(0,1)$ (zur Kontrolle: **qnorm**(0.95)). Verwenden Sie hierzu folgende Verfahren:

(a) Bisektionsverfahren

(b) Newton-Verfahren

Hinweis: Der $k+1$-te Iterationsschritt im Newton-Verfahren lautet

$$x_{0,k+1} = x_{0,k} - \frac{F(x_{0,k}) - 0.95}{F'(x_{0,k})}$$

wobei $F'(x) =$**dnorm**(x).

(c) Numerische Invertierung der Funktion $F(x)$

Lösungsvorschlag:

- `.pdf`-Version: L.12.1
- `.R`-Version: Blatt 12, Aufgabe 1
- `.R`-Version (www): Blatt 12, Aufgabe 1

A.12.2 Schätzung eines eindimensionalen Parameters

Die Daten im `truncpois` Datensatz sind Realisationen einer an der Null abgeschnittenen Poisson-Variable Y, d.h. einer Zufallsvariable mit Wahrscheinlichkeitsfunktion

$$P(Y = y) = \frac{e^{-\lambda}\lambda^y}{(1 - e^{-\lambda})y!} \qquad y = 1, 2, \ldots$$

und Erwartungswert

$$\mathrm{E}(Y) = \frac{\lambda}{1 - e^{-\lambda}}$$

(a) **(freiwillig)** Leiten Sie die Bestimmungsgleichung für den Maximum-Likelihood-Schätzer $\hat{\lambda}$ in dieser Situation her — einmal als Maximierungsproblem, einmal als Nullstellenproblem.
Kontrolle:

- Maximierungsproblem:

$$\left(\frac{1}{e^{\lambda} - 1}\right)^n \frac{\lambda^{n\bar{Y}}}{\prod_{i=1}^{n} x_i!} = \max_{\lambda \in \mathbb{R}}!$$

- Nullstellenproblem:

$$\lambda - \bar{Y}(1 - e^{-\lambda}) = 0$$

mit $\bar{Y}$ dem arithmetischen Mittel der Realisationen von Y

(b) Schreiben Sie eine Funktion, welche für eine gegebene Stichprobe den Maximum-Likelihood-Schätzer als Maximierungsproblem mit einer vorgegebenen Genauigkeit numerisch ermittelt; verwenden Sie einmal eine selbst geschriebene Gittersuche, einmal **optim()** oder **optimize()**.

(c) Schreiben Sie eine Funktion, welche für eine gegebene Stichprobe den Maximum-Likelihood-Schätzer als Nullstellenproblem mit einer vorgegebenen Genauigkeit numerisch ermittelt; verwenden Sie einmal einen selbst geschriebenen Bisektionsalgorithmus, einmal ein ein selbst geschriebenes Newtonverfahren, einmal **uniroot()**.

Kontrolle: Die Bestimmungsgleichung im Newtonverfahren lautet

$$\hat{\lambda}_{m+1} = \hat{\lambda}_m - \frac{\hat{\lambda}_m - \bar{Y}(1 - e^{-\hat{\lambda}_m})}{1 - \bar{Y}e^{-\hat{\lambda}_m}}$$

(d) Integrieren Sie in Ihre Funktionen einen Übergabeparameter, der die Ausgabe des Zwischenergebnisses für $\hat{\lambda}$ am Ende jedes Iterationsschrittes steuert. Realisieren Sie die folgenden Möglichkeiten:

- Ausgabe des Zwischenergebnisses [**print**(), **cat**()]
- Anhalten der Programmausführung und Kontrolle der Variablen [**browser**()]
- Keine Ausgabe des Zwischenergebnisses

(e) Welchen Wert erhalten Sie für den Maximum-Likelihood-Schätzer $\hat{\lambda}$ bei einer vorgegebenen Genauigkeit von 10^{-4}?

Lösungsvorschlag:

- `.pdf`-Version: L.12.2
- `.R`-Version: Blatt 12, Aufgabe 2
- `.R`-Version (www): Blatt 12, Aufgabe 2

A.12.3 Numerische Probleme mit dem Coupon-Collector

Betrachten Sie das Coupon-Collector Problem:
Eine Firma legt ihrem Produkt Sammelbilder bei (n verschiedene Coupons). Um Ihnen eine realistische Größenordnung zu geben: Bei der letzten Fußball-WM gab es ein *PANINI*-Sammelalbum mit $n = 576$ Sammelbildern! Nehmen Sie an, dass Sie jedes Bild mit gleicher Wahrscheinlichkeit beim Kauf vorfinden. Die Wahrscheinlichkeit zum ersten Mal bei N ($\geq n$) gekauften Bildern jeweils mindestens 1 Bild von allen n zu besitzen, ergibt sich mit Hilfe der "Siebformel" als

$$P(N=\nu) = \begin{cases} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (1-k/n)^{\nu} & \text{falls } \nu = n \\ \sum_{k=0}^{n-1} (-1)^k \binom{n-1}{k} (1-(k+1)/n)^{\nu-1} & \text{falls } \nu > n \end{cases}$$

mit Erwartungswert

$$\mathrm{E}\, N = n \cdot \left(1 + \sum_{k=1}^{n-1} (-1)^{k-1} \binom{n}{k} (1 - k/n)^n \cdot \frac{1}{k}\right)$$

(a) Aufgrund numerischer Auslöschungseffekte ist die Formel numerisch unzuverlässig für $n \geq 110$. Um einen Eindruck von der Wahrscheinlichkeitsfunktion zu bekommen, werten Sie die Ausdrücke aus für $n = 2, 3, 5, 20, 50, 100$ und plotten Sie die Funktion $\nu \mapsto P(N = \nu)$ im Bereich von $\nu \in [n, 5n]$ und berechnen Sie jeweils auch den Erwartungswert $E_n := \mathrm{E}(N)$ sowie die Größe E_n/n. Was legt dies für $n = 576$ nahe?

(b) Simulieren Sie nun mit Hilfe von `R / S-Plus` und der Funktion **sample()**, wie viele Bilder Sie kaufen müssen, um von jedem Bild mindestens 1 zu besitzen. Wiederholen Sie dies "mehrere" (≥ 10) Male und berechnen Sie Mittelwert und empirische Varianz. Vergleichen Sie Ihre Ergebnisse mit den Ergebnissen aus Teilaufgabe (b).

Zur Orientierung: 10 Durchläufe auf einem `Pentium III` benötigen etwa 9 Minuten.

(c) Fasst man N als $\sum_{r=1}^{n} X_r$ auf, wobei X_r die Zahl der Käufe ist, die nötig sind, um ein Bild zu erhalten, welches sich von den $r-1$ verschiedenen Bildern unterscheidet, die man bereits erhalten hat, so erhält man die Darstellung als $\mathrm{E}\, N = n \sum_{r=1}^{n} \frac{1}{r}$. Die X_r besitzen dabei eine geometrische Verteilung mit sich ändernder Erfolgswahrscheinlichkeit. Offenbar ist diese Formel numerisch viel zuverlässiger! Berechnen Sie für $n = 10, 20, 50, 100$ den Erwartungswert exakt und näherungsweise, indem Sie die Approximation $\sum_{r=1}^{n} \frac{1}{r} = \log n + \gamma + \epsilon_n$ verwenden, wobei $\epsilon_n \to 0$ und $\gamma \approx 0.5772$ die Eulersche Konstante ist.

(d) Simulieren Sie die geometrisch verteilten Variablen X_r!

Lösungsvorschlag:

- `.pdf`-Version: L.12.3
- `.R`-Version: Blatt 12, Aufgabe 3
- `.R`-Version (www): Blatt 12, Aufgabe 3

A.12.4 Optimale Prognose

Betrachten Sie das Modell $Y = X + \varepsilon$, wobei X und ε stochastisch unabhängige Zufallsvariablen seien. Gesucht ist eine Schätzung für X auf der Basis von Y. Nun sei $X \sim \mathcal{N}(0,1)$ mit Dichte $f_X(x) = \texttt{dnorm}(\texttt{x})$ und

(i) $\varepsilon \sim \mathcal{N}(0,1)$, mit Dichte $f_\varepsilon(\varepsilon) = \texttt{dnorm}(\varepsilon)$

(ii) $\varepsilon \sim 0.9 \cdot \mathcal{N}(0,1) + 0.1 \cdot \mathcal{N}(0,9)$, mit Dichte
$f_\varepsilon(\varepsilon) = 0.9 \cdot \texttt{dnorm}(\varepsilon) + 0.1 \cdot \texttt{dnorm}(\varepsilon, sd = 3)$

(iii) $\varepsilon \sim \text{Cauchy}(0,1)$, mit Dichte $f_\varepsilon(\varepsilon) = \texttt{dcauchy}(\varepsilon)$

Führen Sie für die Fälle (i)-(iii) die folgenden Berechnungen durch:

(a) Berechnen Sie den *bedingten Erwartungswert* $\mathrm{E}(X \,|\, Y = y)$,

wobei

$$\mathrm{E}(X \mid Y = y) = \frac{\int_{-\infty}^{\infty} x f_\varepsilon(y-x) f_x(x)\, dx}{\int_{-\infty}^{\infty} f_\varepsilon(y-x) f_x(x)\, dx} \qquad \text{(A.12.1)}$$

auf einem y-Gitter ($y \in [0, 5]$), indem Sie für jeden y-Wert die Integrale in (A.12.1) mit Hilfe der Funktion $integrate()$ berechnen.

Hinweis: Verwenden Sie die Funktion `bed.Erw()`. Um diese im Fall (ii) anwenden zu können, müssen Sie die Dichte von ε als eine eigene Funktion implementieren.

(b) Berechnen Sie den *Posterior Modus* $\mathrm{postmod}\,(X)$, wobei

$$\mathrm{postmod}\,(X) = \underset{x \in [-5,5]}{\mathrm{argmax}} \frac{f_\varepsilon(y-x) f_x(x)}{\int_{-\infty}^{\infty} f_\varepsilon(y-x) f_x(x)\, dx} \qquad \text{(A.12.2)}$$

auf einem y-Gitter ($y \in [0, 5]$), wobei Sie lediglich den Zähler

von (A.12.2) maximieren. Der Nenner nämlich ist konstant in x und kann daher für die Maximierung vernachlässigt werden. Bestimmen Sie die Maximalstelle, indem Sie den Zähler von $\mathrm{postmod}\,(X)$ auf einem vorgegebenem x-Gitter ($x \in [-5, 5]$) für jedes der y maximieren.

(c) Glätten Sie die Funktionen $y \mapsto \mathrm{E}(X \,|\, Y = y)$ und $y \mapsto \mathrm{postmod}(X)$ mit Hilfe von Splines und plotten Sie die Ergebnisse.

(d) Simulieren Sie $n = 5000$ Realisationen der Zufallsvariablen X und ε und berechnen Sie $y_j = x_j + \varepsilon_j$ $(j = 1, \ldots, n)$ für die Fälle (i)-(iii). Für jede der Stichproben bestimmen Sie dann mit Hilfe der 6 geglätteten Funktionen aus Teil (c) und Inter- bzw. Extrapolationsverfahren den bedingten Erwartungswert und den Posterior Modus für $j = 1, \ldots, n$, d.h. Sie erhalten 18 verschiedene Ergebnisse.

(e) Vergleichen Sie die verschiedenen Schätzungen für X, indem Sie

den empirischen *mean squared error* berechnen, d.h. $\texttt{mean}\,(|\texttt{X}_\texttt{j} - \texttt{k}_\texttt{j}|^2)$, wobei k_j die Ergebnisse aus Teil (d) sind. Stellen Sie die Ergebnisse in einem `Data.frame` zusammen.

Lösungsvorschlag:

- `.pdf`-Version: L.12.4
- `.R`-Version: Blatt 12, Aufgabe 4
- `.R`-Version (www): Blatt 12, Aufgabe 4

A.13 Blatt 13

- Stoff bis einschließlich Abschnitt 7.1.1 (g) — Modellwahl

A.13.1 Lineare Regression

Plotten Sie den `crabs.data` Datensatz, d.h. plotten Sie die Variablen `presz` und `postsz` gegeneinander. Welches lineare Modell vermuten Sie? Geben Sie die entsprechende R-Formel hierfür an.

Lösungsvorschlag:

- `.pdf`-Version: L.13.1
- `.R`-Version: Blatt 13, Aufgabe 1
- `.R`-Version (www): Blatt 13, Aufgabe 1

A.13.2 Freier Fall eines Körpers

Unter Vernachlässigung der Reibung ist die Höhe h eines frei fallenden Körpers zum Zeitpunkt t gegeben durch die Formel

$$h = \beta_0 + \beta_1 t + \beta_2 t^2,$$

wobei β_0 die Höhe des Körpers zum Zeitpunkt $t = 0$ ist, β_1 die Anfangsgeschwindigkeit des Körpers und β_2 die halbe Gravitationsbeschleunigung g.

Es liegt die folgende Serie annähernd gleich genauer Messungen während eines Fallvorgangs vor:

t [s]	h [cm]	t [s]	h [cm]	t [s]	h [cm]	t [s]	h [cm]	t [s]	h [cm]
$\frac{1}{30}$	11.86	$\frac{4}{30}$	26.69	$\frac{7}{30}$	51.13	$\frac{10}{30}$	85.44	$\frac{13}{30}$	129.54
$\frac{2}{30}$	15.67	$\frac{5}{30}$	33.71	$\frac{8}{30}$	61.49	$\frac{11}{30}$	99.08	$\frac{14}{30}$	146.48
$\frac{3}{30}$	20.60	$\frac{6}{30}$	41.93	$\frac{9}{30}$	72.90	$\frac{12}{30}$	113.77		

Schätzen Sie die Konstanten β_0, β_1, β_2 mit Hilfe der Methode der kleinsten Quadrate und versehen Sie Ihre Schätzung der

Gravitationskonstante mit einem Schätzfehler. **Lösungsvorschlag:**

- `.pdf`-Version: L.13.2
- `.R`-Version: Blatt 13, Aufgabe 2
- `.R`-Version (www): Blatt 13, Aufgabe 2

A.13.3 Länge der alten Meile

Entlang einer jahrhundertealten Straße befinden sich sechs Meilensteine an den folgenden Positionen:

Meilenstein 1	784 m
Meilenstein 2	2 460 m
Meilenstein 3	4 147 m
Meilenstein 4	5 826 m
Meilenstein 5	7 515 m
Meilenstein 6	9 187 m

Der Nullpunkt ist dabei beliebig gewählt. Schätzen Sie die Länge der diesen Meilensteinen zugrundeliegenden alten Meile unter Annahme folgender Modelle:

- *Modell 1.* Die Steine wurden einst exakt an den richtigen Positionen eingesetzt, doch im Laufe der Zeit verrückt (beispielsweise ausgepflügt und wieder eingesetzt).

- *Modell 2.* Die Positionen der Steine wurden einst etwas ungenau bestimmt und blieben seither unverändert.

Bestimmen Sie für jedes der beiden Modelle den besten linearen erwartungstreuen Schätzer der Länge der alten Meile, und versehen Sie beide Schätzungen mit einem Standardfehler.

Lösungsvorschlag:

- `.pdf`-Version: L.13.3
- `.R`-Version: Blatt 13, Aufgabe 3
- `.R`-Version (www): Blatt 13, Aufgabe 3

A.13.4 Modellanpassung, Modellwahl

Betrachten Sie den Datensatz `model`.

(a) Vergleichen Sie die folgenden Modelle miteinander

(a) $y_i = \beta_0 + \varepsilon$

(b) $y_i = \beta_0 + \beta_1 x_i + \varepsilon$

(c) $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon$

Welches Modell erweist sich mit Hilfe der Funktionen **update**() (forward bzw. backward), **step**() (bzw. **drop1**()) und leaps() (in des Pakets `leaps` enthalten) jeweils als das *beste*?

(b) Plotten Sie die Daten und die mit Modell (ii) (entspricht dem realen Modell) angepasste Kurve.

(c) Die Daten enthalten eine Lücke, d.h. für $x \in [-3, -1]$ liegen keine Daten vor. Bestimmen Sie mit Hilfe der angepassten Kurve aus Teil (b) den Wert für $x = -2$. Legen Sie im Bereich der Lücke ein 95% Konfidenzintervall um die angepasste Kurve.

Hinweis: Verwenden Sie die Funktion **predict**().

(d) Im Datensatz `luecke` finden sich die fehlenden Werte für das Intervall $[-3, -1]$. Ergänzen Sie zur Kontrolle den Plot aus Teil (c) um diese Daten.

Lösungsvorschlag:

- `.pdf`-Version: L.13.4
- `.R`-Version: Blatt 13, Aufgabe 4
- `.R`-Version (www): Blatt 13, Aufgabe 4

A.14 Blatt 14

- Stoff bis einschließlich Abschnitt 7.1.2 — Generalisiert lineare Modelle

A.14.1 ANOVA

Der Absatz von Produkten wird durch die Preispolitik und die Art der Werbung beeinflußt. Für eine Margarinesorte stehen der Marketing-Abteilung eines Konzerns bezüglich Preispolitik (Faktor A) drei Strategien (Faktorstufen), nämlich Niedrig-, Normal- und Hochpreispolitik, und bezüglich der Art der Werbung (Faktor B) zwei Möglichkeiten (Faktorstufen), nämlich Postwurfsendungen und Anzeigenwerbung, zur Verfügung. Zur Untersuchung dieser Optionen hinsichtlich ihrer Wirkung wurden an 10 zufällig ausgewählten Tagen die abgesetzten Mengen der Margarinesorte in 6 Supermärkten erfaßt, wobei sichergestellt wurde, dass die Kunden eines jeden Supermarktes nur mit einer Kombination von Preis- und Werbestrategie konfrontiert wurden. Stellen Sie die Daten in `margarine.txt` geeignet graphisch dar und führen Sie eine ANOVA durch. Berücksichtigen Sie dabei auch eine mögliche Interaktion

zwischen den beiden Faktoren.

Lösungsvorschlag:

- `.pdf`-Version: L.14.1
- `.R`-Version: Blatt 14, Aufgabe 1
- `.R`-Version (www): Blatt 14, Aufgabe 1

Beispiel stammt aus Fahrmeier, L., Hamerle, A.(1984), Multivariate Statistische Verfahren, p. 168

A.14.2 ANOVA II—ohne Lösung

Laden Sie das Paket `car` von *J.Fox* herunter und installieren Sie es sich. Betrachten Sie den Datensatz `Moore`. Was gibt es darüber zu sagen. Modellieren Sie ein vollständiges Design, indem Sie die Variable `conformity` erklären durch die (kategoriellen) Variablen `fcategory` und `partner.status`. Führen Sie eine ANOVA durch. Was können Sie aus den Ergebnissen schließen?

A.14.3 Box–Cox–Transformation I

Betrachten Sie den gauge Datensatz.

(a) Passen Sie an die Daten ein einfaches lineares Modell an und plotten Sie die Residuen. Was fällt auf?

(b) Finden Sie mit Hilfe der Funktion boxcox() aus dem MASS Paket eine geeignete Transformation für die Daten.

(c) Passen Sie an die transformierten Daten ein einfaches lineares Modell an und plotten Sie die Residuen.

(d) Plotten Sie nun die Daten und die angepassten Kurven aus Teil (a) und (c).

Lösungsvorschlag:

- `.pdf`-Version: L.14.3
- `.R`-Version: Blatt 14, Aufgabe 3
- `.R`-Version (www): Blatt 14, Aufgabe 3

A.14.4 Box–Cox–Transformation II

Betrachten Sie noch einmal Aufgabe A.12.3; schließen Sie nun mit Hilfe der Box-Cox-Transformation auf die Asymptotik von $\mathrm{E}[N]$.

Lösungsvorschlag:

- `.pdf`-Version: L.14.4
- `.R`-Version: Blatt 14, Aufgabe 4
- `.R`-Version (www): Blatt 14, Aufgabe 4

A.14.5 Generalisiert lineares Modell

Betrachten Sie den `adver` Datensatz. Es handelt sich dabei um Daten, die im Zusammenhang mit einer Studie über Wirkung und Wahrnehmung der Werbung im Fernsehen erhoben wurden. Es wurde 66 Personen während eines Zeitraumes von 171 Wochen wöchentlich eine Frage zu Werbespots für einen bekannten Schokoladenriegel gestellt. Außerdem wurde noch der wöchentliche Werbeaufwand in einer skalaren, metrischen Größe erfasst. Da für einige Wochen die Antworten leider fehlen, wurden die entsprechenden fehlenden Werte mit "0"kodiert.
Führen Sie eine Analyse mit Hilfe des Logit- und des Probit-Modells durch und vergleichen Sie die Ergebnisse. Verwenden Sie dabei auch geeignete graphische Darstellungen.

Lösungsvorschlag:

- `.pdf`-Version: L.14.5
- `.R`-Version: Blatt 14, Aufgabe 5
- `.R`-Version (www): Blatt 14, Aufgabe 5

A.15 Blatt 15

- Stoff bis einschließlich Abschnitt 7.2 — Elemente multivariater Statistik

A.15.1 Multivariate Normalverteilung

(a) Schreiben Sie selbst eine Funktion zur Erzeugung multivariater normalverteilter Daten zu vorgegebenem Mittelwert und Kovarianz.

(b) Erzeugen Sie mit dieser Funktion bivariate Zufallsvariablen gemäß

$$0.9\mathcal{N}_2\left(0, \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}\right) + 0.1\mathcal{N}_2\left(\begin{pmatrix} 5 \\ -2 \end{pmatrix}, \begin{pmatrix} 0.1 & 1 \\ 1 & 4 \end{pmatrix}\right),$$

wobei die erste Komponente einer idealen Verteilung, die zweite einer Kontamination entspricht, und protokollieren Sie bei der Simulation mit, welche Situation jeweils in einer einzelnen Beobachtung vorliegt.

(c) Erzeugen Sie eine ellipsenförmige 95%-Konfidenzregion für die ideale Verteilung $\mathcal{L}(X_{\mathrm{id}})$.

(d) Ergänzen Sie den Plot nun um die in Teil (b) generierten Daten. Dabei zeichnen Sie die Punkte aus der kontaminierenden

Verteilung rot, falls sie innerhalb der Ellipse liegen und orange, falls sie außerhalb liegen. Die Punkte aus der idealen Verteilung zeichnen Sie grün, falls Sie innerhalb der Ellipse liegen und blau, falls sie außerhalb liegen.

(e) Finden Sie heraus, wie man die Aufgabe mit den Paketen `mvtnorm`, `ellipse` erledigt.

Lösungsvorschlag:

- `.pdf`-Version: L.15.1
- `.R`-Version: Blatt 15, Aufgabe 1
- `.R`-Version (www): Blatt 15, Aufgabe 1

A.15.2 Clustering, Diskriminanzanalyse

Betrachten Sie den Banknoten-Datensatz `bank2`. Der Datensatz enthält Messungen von 100 echten und 100 falschen Schweizer Banknoten. Es wurden dabei 6 verschiedene Größen vermessen (Länge, Breite, Diagonale, ...).

(a) Wenden Sie auf den gesamten `bank2` Datensatz mindestens zwei verschiedene Clusteralgorithmen an. Vergleichen und interpretieren Sie die Ergebnisse.

(b) Wenden Sie eine lineare Diskriminanzanalyse auf den `bank2` Datensatz an. Verwenden Sie die Beobachtungen 1 bis 70 und 131 bis 200 als Trainingsstichprobe und die Beobachtungen 71 bis 130 als Validierungsstichprobe.

Lösungsvorschlag:

- `.pdf`-Version: L.15.2
- `.R`-Version: Blatt 15, Aufgabe 2

- .R-Version (www): Blatt 15, Aufgabe 2

A.15.3 Hauptkomponentenanalyse, Faktoranalyse

Betrachten Sie erneut den `bank2` Datensatz.

(a) Führen Sie für den gesamten `bank2` Datensatz eine Hauptkomponentenanalyse durch und interpretieren Sie das Ergebnis.

(b) Wenden Sie auf den gesamten `bank2` Datensatz eine Faktoranalyse an und interpretieren Sie das Ergebnis.

Lösungsvorschlag:

- `.pdf`-Version: L.15.3
- `.R`-Version: Blatt 15, Aufgabe 3
- `.R`-Version (www): Blatt 15, Aufgabe 3

A.15.4 Hauptkomponentenanalyse, Faktoranalyse II —ohne Lösung

Betrachten Sie erneut den `Plato` Datensatz aus Blatt 8.

(a) Führen Sie für den `Plato` Datensatz eine Hauptkomponentenanalyse durch und interpretieren Sie das Ergebnis.

(b) Wenden Sie auf den `Plato` Datensatz eine Faktoranalyse (mit zwei Faktoren) an und interpretieren Sie das Ergebnis.

A.15.5 Normalisierte Hauptkomponentenanalyse

Beziehen Sie auf der Service–Homepage oder unter

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/SPlus0203/uscrime.dat

den Datensatz `uscrime.dat`:

50 Messungen von 11 Variablen; es wird die Zahl der Verbrechen im Jahr 1985 in jedem der 50 Staaten der USA in verschiedenen Kategorien — `X3`–`X7` angegeben:

`X1`	Fläche des Staates	`X7`	Einbrüche
`X2`	Bevölkerung des Staates 1985	`X8`	Diebstahl (larcery)
`X3`	Morde	`X9`	Autodiebstahl (autothieft)
`X4`	Vergewaltigungen	`X10`	Region des Staates
`X5`	Raubüberfälle	`X11`	GebietsNr. des Staates
`X6`	Anschläge		

Regionen codiert als

1	Nordosten	3	Süden
2	Mittlerer Westen	4	Westen

Gebiete codiert als

1	Neuengland	6	East South Central
2	Mittlere Atlantikküste	7	West South Central
3	East North Central	8	Gebirgsregion
4	West North Central	9	Pazifikküste
5	Südliche Atlantikküste		

Daten nach Härdle, W. and Simar, L. (2003), S. B.18; die Werte liegen als `ASCII`–Text vor (ohne Kopfzeilen!).

Wenden Sie eine *normalisierte Hauptkomponentenanalyse (NPCA)* auf den `uscrime.dat` Datensatz an. Dabei werden alle Variablen vor der eigentlichen Hauptkomponentenanalyse zuerst zentriert und standardisiert. Interpretieren Sie die Ergebnisse. Ist es nötig, die dritte PC zu betrachten? Können Sie Unterschiede zwischen den vier Regionen erkennen? Wiederholen Sie die Analyse ohne die Variable `X1`

Lösungsvorschlag:

- `.pdf`-Version: L.15.5
- `.R`-Version: Blatt 15, Aufgabe 5
- `.R`-Version (www): Blatt 15, Aufgabe 5

A.16 Blatt 16

- Stoff bis einschließlich Abschnitt 7.4.3 — Punktprozesse

A.16.1 Erstellung von Zeitreihenobjekten

Installieren Sie die Pakete `its` und `zoo`.

(a) Beziehen Sie unter `Yahoo finance` einen Datensatz mit Aktienkursen Ihrer Wahl und importieren Sie diesen als einen Data Frame in R.

(b) Wandeln Sie einen vollständigen Teilabschnitt dieses Datensatzes in ein Zeitreihenobjekt der Klasse `ts`.

(c) Wandeln Sie Ihren (möglicherweise mit Missings versehenen) Datensatz in ein Zeitreihenobjekt der Klasse `its`.

(d) Wandeln Sie Ihren (möglicherweise mit Missings versehenen) Datensatz in ein Zeitreihenobjekt der Klasse `zoo`.

In allen Teilaufgaben sollen Sie die "Time Stamps" als Zeitobjekte der Klasse `chron` erstellen.

Lösungsvorschlag:

- `.pdf`-Version: L.16.1
- `.R`-Version: Blatt 16, Aufgabe 1
- `.R`-Version (www): Blatt 16, Aufgabe 1

A.16.2 Zeitreihenanalyse I

Beziehen Sie auf der Service–Homepage oder unter

http://www.uni-bayreuth.de/departments/math/org/mathe7/rkurs/SPlus0203/daten01.txt

den Datensatz `daten01.txt` mit einem Zeitindex und 11 verschiedenen realen und simulierten Zeitreihen der Länge 100

(a) Erstellen sie Plots des Zeitindexes gegen alle anderen Variablen. Welche Zeitreihen erscheinen (visuell) stationär? Bei welchen Zeitreihen liegt deutlich ein Trend vor, bei welchen ist (visuell) die Autokorrelation zum Lag 1 stark positiv?

(b) Lassen Sie sich von allen Zeitreihen die (empirische) Autokovarianzfunktion, Autokorrelationsfunktion und die paritelle Autokorrelationsfunktion ausgeben. Stimmen die Resultate mit den visuellen Eindrücken überein?

(c) Bei welchen der in Teil (a) und (b) sich als nicht stationär erwiesenen Zeitreihen eleminiert Differenzenbildung den Trend?

Lösungsvorschlag:

- `.pdf`-Version: L.16.2
- `.R`-Version: Blatt 16, Aufgabe 2
- `.R`-Version (www): Blatt 16, Aufgabe 2

A.16.3 Zeitreihenanalyse II

(a) Simulieren Sie mit Hilfe von arima.sim() $n = 100$ Beobachtungen aus einem ARMA(1,1)-Modell mit AR-Parameter $\phi = 0.7$, MA-Parameter $\xi = 0.35$ und Innovationen, welche normalverteilt sind mit Erwartungswert 0 und Varianz 0.5. Schätzen Sie dann die Parameter und die Autokovarianzfunktion.

(b) Betrachten Sie den `LakeHuron` Datensatz aus dem per default bereits eingeladenen Paket `stats`. Subtrahieren Sie von den Werten zuerst 570 und von den Ergebnissen dann das arithmetische Mittel dieser *neuen* Werte. Passen Sie nun an diesen zentrierten Datensatz mit Hilfe von arima0() bzw. arima() ein ARMA(1,1) Modell an und schätzen Sie die Autokovarianzfunktion. Berechnen Sie außerdem eine 5-Schritt Prognose.

Lösungsvorschlag:

- `.pdf`-Version: L.16.3
- `.R`-Version: Blatt 16, Aufgabe 3
- `.R`-Version (www): Blatt 16, Aufgabe 3

A.16.4 Kalman-Filter in R —ohne Lösung

Installieren Sie das unter

http://www.uni-bayreuth.de/departments/math/org/mathe7/robKalman

verfügbare Paket `robKalman`.

(a) Simulieren Sie mit `simulateState` und `simulateObs` jeweils Zeitreihen der Länge 200 aus folgendem, zeitinvarianten Zustandsraummodell:

$$a_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \Sigma_0 = 0 \in \mathbb{R}^{2\times 2}, \quad F = \begin{pmatrix} .7 & .5 \\ .2 & 0 \end{pmatrix}, \quad Q = \begin{pmatrix} 2 & .5 \\ .5 & 1 \end{pmatrix},$$

$$Z = (1,\ -.5)\,, \qquad V = 1$$

(b) Rekonstruieren Sie mithilfe des Kalman-Filters `KalmanFilter` die Zustände x_t bestmöglich.

(c) Wandeln Sie sowohl die Folge der simulierten Zustände als auch die der gefilterten Werte in Objekte vom Typ `ts` und plotten Sie diese gemeinsam.

A.16.5 Räumliche Statistik

Wiederholen Sie mit dem `s101` Datensatz das Beispiel zur räumlichen Statistik aus der Vorlesung. Es handelt sich dabei um ein Objekt der Klasse `geodata`. Verwenden Sie daher zum Laden des Datensatzes den Befehl `load()`!

Hinweis: Die Befehle finden sich im R Skript `geoRintro.R`.

Lösungsvorschlag:

- `.pdf`-Version: L.16.4
- `.R`-Version: Blatt 16, Aufgabe 5
- `.R`-Version (www): Blatt 16, Aufgabe 5

A.17 Blatt 17

- Stoff bis einschließlich Abschnitt 8.1 — Objektorientierung in R

A.17.1 Entwurf einer Simulationsklasse

(a) Entwerfen Sie eine Simulationsklasse; diese sollte als Slot die Daten enthalten, sowie den Stichprobenumfang und die Zahl der Runs, den Seed und einen Filenamen in dem die Informationen zur Erzeugung der Simulation abgelegt werden. Als Methoden sollte die Funktion, mit der gesampelt werden soll, eine Ausgabe vom Typ `summary` sowie eine `plot`-Methode angebunden sein.

(b) Leiten Sie nun eine Klasse ab, bei der Sie geeignet kontaminierte / und ideal verteilte Stichproben ziehen können und dabei geeignet mitprotokollieren, welche Daten kontaminiert sind und welche nicht.

Lösungsvorschlag:

- `.pdf`-Version: L.17.1
- `.R`-Version: Blatt 17, Aufgabe 1
- `.R`-Version (www): Blatt 17, Aufgabe 1

A.17.2 Indexoperator

Definieren Sie einen Indexoperator für die Klasse Simulation, so dass bei einem Objekt X dieser Klasse `X[i]` die i-te Stichprobe (i-ten Run) zurückgibt.

Lösungsvorschlag:

- `.pdf`-Version: L.17.2
- `.R`-Version: Blatt 17, Aufgabe 2
- `.R`-Version (www): Blatt 17, Aufgabe 2

A.17.3 Vorbereitungen zur Verteilungsklasse —ohne Lösung

(a) Legen Sie eine virtuelle Klasse `Parameter` an —mit einzigem Slot `Name`—, aus der Sie eine Klasse für die beiden Parameter einer multivariaten Normalverteilung ableiten —mit entsprechenden Accessor– und Replacement–Funktionen—, und von dieser wiederum den Spezialfall der univariaten Normalverteilung.

(b) Legen Sie eine virtuelle Klasse `rSpace` an, die den Realisationsraum der Zufallsvariablen aus der Verteilung modellieren soll. Diese sollte nur einen Slot `Name` haben; als abgeleitete Klasse definieren Sie den Euklidischen Raum $\mathbb{R}^p$, indem Sie ihm einen Slot `Dimension` und eine Methode `liesIn` spendieren, wobei letztere `TRUE/FALSE` antwortet, je nachdem ob die Dimensionalität passt.

A.17.4 Verteilungsklasse II

(a) Generieren Sie eine virtuelle Verteilungsmutterklasse mit Slots vom Typ `Parameter` und vom Typ `rSpace`, sowie den Methoden für "`r`", "`d`", "`p`" und "`q`", von der dann die Normalverteilung ein Spezialfall ist.

(b) Machen Sie bei der Verteilungsklasse die Operationen

$$+: \quad \texttt{Verteilung} \times \texttt{numeric} \to \texttt{Verteilung},$$
$$(F, x) \mapsto \text{“}F + x'' = F(\cdot - x)$$
$$* \quad : \texttt{Verteilung} \times \texttt{numeric} \to \texttt{Verteilung},$$
$$(F, x) \mapsto \text{“}F * x'' = F(\cdot / x)$$

verfügbar, so dass, falls `X` eine Instanz aus dieser Klasse ist, Ausdrücke wie `2*X+3` einen Sinn ergeben.

(c) Schreiben Sie nicht notwendig optimale default–Verfahren, um mithilfe der "r"–Methode einer Verteilung die "d"–, "p"– und "q"–Methoden zu rekonstruieren

(d) Machen Sie die Gruppe der `Math`–Methoden für die Klasse verfügbar, so dass, falls `X` eine Instanz aus dieser Klasse ist, Ausdrücke wie `Y<-2*sin(X*exp(X))+3` ein Sinn ergeben. Gehen Sie dabei von der "`r`"–Methode der Klasse `X` aus und beginnen Sie mit der "`r`"–Methode für `Y`. Verwenden Sie dann Teil (c).

(e) Schreiben Sie für den univariaten Fall Methoden `m1df` und `m2df` für die folgenden beiden Funktionen

$$\texttt{m1df}(t) \quad := \quad \mathrm{E}[X\ \mathrm{I}_{\{X\leq t\}}] = \int_{(-\infty;t]} x\, P(dx) \quad \text{(A.17.1)}$$

$$\texttt{m2df}(t) \quad := \quad \mathrm{E}[X^2\ \mathrm{I}_{\{X\leq t\}}] = \int_{(-\infty;t]} x^2\, P(dx) \quad \text{(A.17.2)}$$

Schreiben Sie diese einmal mit `integrate` und der "`d`–"Methode, und spezialisieren Sie die Funktion durch Überladen für die Normalverteilung.

A.17.5 Simulationsklasse II

(a) Machen Sie die Gruppe der `Math`–Methoden für die Klasse verfügbar, so dass, falls `X` eine Instanz aus dieser Klasse ist, Ausdrücke wie `Y<-2*X+3` einen Sinn ergeben, indem diese Elementweise auf den/die Datenslots wirken.

(b) Schreiben Sie `save`– und `load`–Methoden für diese Klasse, das alles, aber nicht die Daten abspeichert.

Lösungsvorschlag:

- `.pdf`-Version: L.17.5
- `.R`-Version: Blatt 17, Aufgabe 5
- `.R`-Version (www): Blatt 17, Aufgabe 5

A.17.6 Schätzerauswertungsklasse

(a) Definieren Sie eine Klasse `DatenClass` als Oberklasse von `Simulation`. Diese sollte die Slots `Name`, `samplesize`, `Daten`, `runzahl` enthalten. Damit könnte man dann auch reale Daten abdecken — `runzahl` dann als Versuchswiederholung interpretiert.

(b) Definieren Sie eine Klasse `AuswertungsClass` mit Slots `Schätzer` — eine Funktion

$$\texttt{Schätzer} : \texttt{Datenclass} \longrightarrow \texttt{numeric}$$

die den Schätzer auf den Daten auswertet, `DatenName` — der Name des Datensatzes, und `Ergebnis` — das Ergebnis der Schätzerauswertung auf den Daten.

(c) Schreiben eine Funktion `Auswertung`,

$$\texttt{Werteaus} : \texttt{DatenClass} \times \texttt{Schätzer} \longrightarrow \texttt{SchätzerClass}$$

die eine Instanz vom Typ `AuswertungsClass` erzeugt, wobei sie die Slots `Schätzer`, `DatenName` und `Ergebnis` belegt.

(d) Füllen Sie als *proof of concept* mit dem Median als Schätzer eine entsprechende Instanz vom Typ `Ergebnis` mit der Funktion `Auswertung`.

Lösungsvorschlag:

- `.pdf`-Version: L.17.6
- `.R`-Version: Blatt 17, Aufgabe 6
- `.R`-Version (www): Blatt 17, Aufgabe 6

A.18 Blatt 18

- Stoff bis einschließlich Abschnitt 8.2 — eigene Pakete in R

A.18.1 Checken/Erstellen eines Pakets

(a) Bereiten Sie ggf. Ihren Rechner wie in Abschnitt 8.2.12 beschrieben auf die Anlage eines R-Pakets vor.

(b) Legen Sie ein Directory `Rtest` an. Laden Sie sich auf Ihren Rechner die `tar.gz`-Version des Pakets `distr` unter http://cran.us.r-project.org/src/contrib/Descriptions/distr.html. Entpacken Sie dieses Archiv in einem Unter-Directory `Rtest/mdistr`.

(c) Modifizieren Sie den Titel des Pakets im `DESCRIPTION`-File auf `mdistr`.

(d) Checken Sie das Paket mit `R CMD check` und erstellen Sie es mit `R CMD build` als `tar.gz` bzw. als Windows-Binary (`.zip`-File).

(e) Installieren Sie das neue Paket `mdistr` und testen Sie es mit den Demos.

(f) Deinstallieren Sie das Paket `mdistr` wieder.

Kopieren Sie dabei alle Konsolenergebnisse in eine Textdatei und schnüren Sie diese mit der `tar.gz`– und der `zip`–Version Ihres Paketes in ein `zip`-File.

Lösungsvorschlag:

hier liegt noch keine Musterlösung vor.

A.18.2 Anlegen eines Daten-Pakets

Erstellen Sie aus dem Datensatz `hills` aus dem `MASS`-Paket ein neues Paket

(a) Lassen Sie sich mit `package.skeleton` eine Hülle erstellen.

(b) Legen Sie ein `DESCRIPTION`-File an.

(c) Erstellen Sie ein Dokumentations- / `rd`-File

(d) Checken und erstellen Sie das Paket (ungepackt und als Binary).

(e) Installieren und deinstallieren Sie es.

Kopieren Sie dabei alle Konsolenergebnisse in eine Textdatei und schnüren Sie diese mit der `tar.gz`– und der `zip`–Version Ihres Paketes in ein `zip`-File.

Lösungsvorschlag:

hier liegt noch keine Musterlösung vor.

A.18.3 Anlegen eines eigenen R-Pakets

Fassen Sie alle Ihre Routinen zu Blatt A.17 in einem R–Paket zusammen. Gehen Sie vor wie bei Aufgabe 1 und erstellen Sie die notwendigen `DESCRIPTION`-Files und `rd`-Files. Erstellen Sie ein `NAMESPACE`-File. Checken und erstellen Sie das Paket (ungepackt und als Binary). Kopieren Sie dabei alle Konsolenergebnisse in eine Textdatei und schnüren Sie diese mit der `tar.gz`– und der `zip`–Version Ihres Paketes in ein `zip`-File.

Lösungsvorschlag:

hier liegt noch keine Musterlösung vor.

A.18.4 Arbeit mit `Sweave`

Führen Sie mit der Schätzerklasse aus Blatt A.17 eine kleine Simulationsstudie durch und beschreiben die Ergebnisse in einem LaTeX-File, das Sie mit `R-Code`–Chunks versehen. Erstellen Sie jeweils ein `.R`-, ein `.tex`- und ein `.txt`-File (letzteres mit den Befehlen zur Erzeugung der Vignette). Schnüren Sie alle Files in ein `zip`-File.

Lösungsvorschlag:

hier liegt noch keine Musterlösung vor.

A.19 Blatt 19

- Stoff bis Ende des Kurses

A.19.1 R und MySQL

(a) Installieren Sie MySQL und RMySQL.

(b) Erzeugen Sie bei sich die Sakila Datenbank
http://dev.mysql.com/doc/sakila/en/sakila.html.

(c) Finden Sie mit SQL-Anfragen die überfälligen DVDs heraus und importieren Sie das Ergebnis dieser Anfrage nach R.

Erstellen Sie jeweils ein .txt- und ein .R-File (ersteres mit den SQL-Anfragen. Schnüren Sie alle Files in ein zip-File.

Lösungsvorschlag:
hier liegt noch keine Musterlösung vor.

A.19.2 Aufruf von C Code unter R

(a) Bereiten Sie Ihren Rechner wie in Abschnitt 8.3.8 beschrieben auf die Anlage eines R-Pakets vor.

(b) Schreiben Sie den Algorithmus aus Beispiel 3.6-2 um in C.

(c) Schreiben Sie eine (einfache) R Funktion, die Ihre C-Funktion aus (b) mit mit der .C-Schnittstelle ansteuert.

(d) Wiederholen Sie (c) nur jetzt mit der .Call-Schnittstelle

Erstellen Sie jeweils ein .c-, ein .R- und ein .txt-File (letzteres mit den Befehlen zur Erzeugung des .so/.dll-Files). Schnüren Sie alle Files in ein zip-File.

Lösungsvorschlag:
hier liegt noch keine Musterlösung vor.

A.19.3 Aufruf von Fortran Code unter R

(a) Laden Sie sich aus dem Netz den Fortran-Quelltext zur Integrations-Routine dqagse aus der Fortran-Bibliothek quadpack herunter.

(b) Verwenden Sie dqagse, um das gleiche wie in Aufgabe A.19.3 mit der .Fortran-Schnittstelle zu erreichen.

Erstellen Sie ein .R- und ein .txt-File (letzteres mit den Befehlen zur Erzeugung des .so/.dll-Files). Schnüren Sie alle Files in ein zip-File.

Lösungsvorschlag:
hier liegt noch keine Musterlösung vor.

A.19.4 Paralleles Rechnen mit R

(a) Verschaffen Sie sich Zugang zu einem Compute-Cluster mit bereits installiertem R ;-)

(b) Richten Sie sich eine lokale Library ~/myRlibs ein, in die Sie die Pakete `snow` und `Rmpi` aus den Sourcen mit `R CMD INSTALL` installieren, und erstellen Sie in Ihrem Stammverzeichnis ein File `.Renviron` mit Inhalt R_LIBS="~/myRlibs:${R_LIBS}".

(c) Wiederholen Sie Aufgabe A.5.1, allerdings diesmal mit $M = 50000$ Stichproben der Länge n, für $n = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100, 200, 300, 400, 500, 1000$ —auf mindestens 19 Knoten.

(d) Versuchen Sie mit boxcox das asymptotische Verhalten der Verteilungsfunktion in den Flanken in Abhängigkeit von n zu beschreiben.

Erstellen Sie jeweils ein `.c`-, ein `.R`- und ein `.txt`-File (letzteres mit

den Befehlen zur Erzeugung des `.so`/`.dll`-Files). Schnüren Sie alle Files in ein `zip`-File.

Lösungsvorschlag:

hier liegt noch keine Musterlösung vor.

L Lösungen

L.1 Lösungsvorschläge Blatt 1

L.1.1 Arbeit mit R-Skripten am Beispiel eines beliebigen editors: (ohne direkte Anbindung an R)

(a) Öffnen Sie eine neue Text-Datei – Benennen und speichern Sie die Datei mit der Dateiendung `.R` – Starten Sie R

(b) Laden und Abarbeiten der kompletten (gespeicherten) Datei mit **source**(**file** ="c:/.../bsp.R", ...) , bzw.
eval(**parse**(**file** ="c:/.../bsp.R", n=−1, ...))

(c) Laden und Abarbeiten der ersten m Befehle mit
eval(**parse**(**file** ="c:/.../bsp.R", n=m, ...))

(d) Laden und Abarbeiten Sie einzelner Befehle (am besten) mit *cut and paste*.

L.1.2 Auffinden von Datensätzen

(a) Zum Beispiel: http://www.oanda.com/convert/fxhistory?lang=de

(b) Zum Beispiel:

http://www.dwd.de/research/klis/daten/online/wwr/form_deu.htm

(c) Aufruf von **library** () liefert:
In S-Plus 2000:

```
The following sections are available in the library directory:
SECTION     BRIEF DESCRIPTION
chron       Functions to handle dates and times.
class       Functions for non-parametric classification from
            Venables and Ripley.
design      Functions for biostats and epidemiological
            modeling from Frank Harrel.
examples    Functions and objects from The New S Language.
hmisc       Miscellaneous functions from Frank Harrel.
maps        Display of maps with projections.
MASS        Functions and data sets from "Modern Applied
            Statistics with S-PLUS" by Venables and Ripley.
Matrix      New Matrix class functions for numerical linear
```


```
              algebra
nlme2         Non-Linear Mixed Effects Functions version 2.1
nnet          Software for feed-forward neural networks from
              Venables and Ripley.
olddates      Date functions which use deprecated century
              handling rules.
progdraw      Sdraw example from Programmer's Manual.
progexam      Examples from Programmer's Manual.
semantics     Functions from chapter 11 of The New S Language.
spatial       Spatial statistics library from Venables
              and Ripley.
For more information on each library section see the README
file in each library section directory.  You can also in
S-PLUS run: \lstinline{library( help = <section_name> )}
```

In R 1.6.0:

```
Packages in library 'C:/PROGRAMME/R/RW1060/library':

acepack           ace() and avas() for selecting regression
                  transformations
adapt             adapt -- multidimensional numerical
                  integration
```


```
agce            analysis of growth curve experiments
akima           Interpolation of irregularly spaced data
AnalyzeFMRI     Functions for analysis of fMRI datasets
                stored in the ANALYZE format.
ape             Analyses of Phylogenetics and Evolution
ash             David Scott's ASH routines
base            The R base package
Bhat            General likelihood exploration
bindata         Generation of Artificial Binary Data
blighty         British Isles coastlines
boot            Bootstrap R (S-Plus) Functions (Canty)
bootstrap       Functions for the Book "An Introduction
                to the Bootstrap"
bqtl            Bayesian QTL mapping toolkit
brlr            Bias-reduced Logistic Regression
car             Companion to Applied Regression
cclust          Convex Clustering Methods and Clustering
                Indexes
cfa             Analysis of configuration frequencies (CFA)
chron           Chronological objects which can handle
                dates and times
CircStats       Circular Statistics
```


```
class          Functions for classification
cluster        Functions for clustering (by Rousseeuw et al.)
cmprsk         Subdistribution Analysis of Competing Risks
cobs           COBS -- Constrained B-splines
CoCoAn         Constrained Correspondence Analysis
coda           Output analysis and diagnostics for MCMC
combinat       combinatorics utilities
conf.design    Construction of factorial designs
cramer         Multivariate nonparametric Cramer-Test
ctest          Classical Tests
date           Functions for handling dates
dblcens        Compute the NPMLE of distribution from doubly
               censored data
deal           Learning Bayesian Networks with Mixed
               Variables
deldir         Delaunay Triangulation and Dirichlet
               (Voronoi) Tesselation.
Devore5        Data sets from Devore's "Prob and Stat
               for Eng (5th ed)"
diamonds       Analysis and sampling grids from diamond
               partitions
dichromat      Color schemes for dichromats
```


dr	Dimension reduction for regression
dse1	Dynamic Systems Estimation (time series package)
dse2	Dynamic Systems Estimation - extensions
e1071	Misc Functions of the Department of Statistics (e1071), TU Wien
eda	Exploratory Data Analysis
ellipse	Functions to plot pairwise confidence regions
emplik	empirical likelihood ratio for censored/truncated data
EMV	Estimation of Missing Values for a Data Matrix
evd	Functions for extreme value distributions
exactRankTests	Exact Distributions for Rank and Permutation Tests
fastICA	FastICA algorithms to perform ICA and Projection Pursuit
fdim	Functions for calculating fractal dimension
fields	Tools for spatial data
foreign	Read data stored by Minitab, SAS, SPSS, ...
fracdiff	Fractionally differenced ARIMA (p,d,q) models

g.data	Delayed-Data Packages
gafit	Genetic Algorithm for Curve Fitting
gee	Generalized Estimation Equation solver
geepack	Generalized Estimating Equation Package
GeneSOM	Clustering Genes using Self-Organizing Map
GenKern	Functions for generating and manipulating generalised kernel density estimates
geoR	geoR - functions for geostatistical analysis
geoRglm	geoRglm a package for generalised linear spatial models
gld	Basic functions for the generalised (Tukey) lambda distribution
gllm	Generalised log-linear model
GLMMGibbs	Generalised Linear Mixed Models by Gibbs Sampling
gregmisc	Greg's Miscellaneous Functions
grid	The Grid Graphics Package
gss	General Smoothing Splines
haplo.score	Score Tests for Association of Traits with Haplotypes when Linkage Phase is Ambiguous.
ifs	Iterated Function Systems estimator
ineq	Measuring inequality, concentration and

	poverty
ipred	Improved Predictors
ISwR	Data sets for "Introductory Statistics with R"
KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)
KMsurv	Data sets from Klein and Moesschberger (1997), "Survival Analysis"
knnTree	k-nn classification with variable selection inside leaves of a tree
lasso2	L1 constrained estimation aka 'lasso'
lattice	Lattice Graphics
leaps	regression subset selection
lgtdl	A set of methods for longitudinal data objects.
lmtest	Testing Linear Regression Models
logspline	Logspline density estimation
lokern	Kernel Regression Smoothing with Local or Global Plug-in Bandwidth
lpridge	Local Polynomial (Ridge) Regression
lqs	Resistant Regression and Covariance Estimation
maptree	Mapping, pruning, and graphing tree models

MASS	Main Library of Venables and Ripley's MASS
Matrix	A Matrix package for R
maxstat	Maximally Selected Rank- and Gauss statistics
mda	Mixture and flexible discriminant analysis
meanscore	Meanscore method for missing covariate data in logistic regression models
methods	Formal Methods and Classes
mgcv	Multiple smoothing parameter estimation and GAMs by GCV
mlbench	Machine Learning Benchmark Problems
moc	General Nonlinear Mixture of curves.
modreg	Modern Regression: Smoothing and Local Methods
muhaz	Hazard Function Estimation in Survival Analysis
multcomp	Multiple Tests and Simultaneous Confidence Intervals
multiv	Multivariate Data Analysis Routines
mva	Classical Multivariate Analysis
mvnmle	ML estimation for multivariate normal data with missing values.
mvtnorm	Multivariate Normal and T Distribution

netCDF	read data in UCAR's netCDF format
NISTnls	Nonlinear least squares examples from NIST
nlme	Linear and nonlinear mixed effects models
nlrq	Nonlinear quantile regression
nls	Nonlinear regression
nnet	Feed-forward neural networks and multinomial log-linear models
norm	Analysis of multivariate normal datasets with missing values
normix	Normal Mixture Models (1-d) {Classes and Methods}
npmc	Nonparametric Multiple Comparisons
Oarray	Arrays with arbitrary offsets
odesolve	Solvers for Ordinary Differential Equations
oz	Plot the Australian coastline and states
panel	Panel
pastecs	Package for Analysis of Space-Time Ecological Series
pcurve	Principal Curve analysis
pear	periodic autoregression library
permax	permax
pinktoe	convert S trees to HTML/perl for interactive

	tree traversal
pixmap	Bitmap Images (‘‘Pixel Maps’’)
polymars	Polychotomous Regression based on MARS
polynom	A collection of functions to implement a class for univariate polynomial manipulations
princurve	Fits a Principal Curve in Arbitrary Dimension
pspline	Penalized Smoothing Splines
PTAk	Principal Tensor Analysis on k modes
qtl	Tools for analyzing QTL experiments
quadprog	Functions to solve Quadratic Programming Problems.
quantreg	Quantile Regression
qvcalc	Quasi-variances for Model Coefficients
RadioSonde	Tools for plotting skew-T diagrams and winds profiles
RandomFields	Simulation and Analysis of Random Fields
randomForest	Breiman’s random forest for classification and regression
RArcInfo	Functions to import data from Arc/Info V7.x binary coverages
relimp	Relative Contribution of Effects in a Regression Model

rmeta	Meta-analysis
RMySQL	MySQL interface for R
rpart	Recursive partitioning
Rwave	Time-Frequency analysis of 1-D signals
SASmixed	Data sets from "SAS System for Mixed Models"
scatterplot3d	3D Scatter Plot
sem	Structural Equation Models
serialize	Simple Serialization Interface
sgeostat	An Object-oriented Framework for Geostatistical Modeling in S+
sm	kernel smoothing methods: Bowman & Azzalini (1997)
sma	Statistics for Microarray Analysis
sn	The skew-normal and skew-t distributions
sna	Tools for Social Network Analysis
sound	A Sound Interface for R
spatial	functions for kriging and point pattern analysis
spatstat	Analysis of spatial point patterns
spdep	Spatial dependence: weighting schemes, statistics and models
splancs	Spatial and Space-Time Point Pattern Analysis

splines	Regression Spline Functions and Classes
StatDataML	alpha implementation of the StatDataML proposal
stepfun	Step Functions, including Empirical Distributions
strucchange	Testing for Structural Change
subselect	Selecting variable subsets.
SuppDists	Supplementary distributions
survival	Survival analysis, including penalised likelihood.
syskern	Coding Kernel for R/S Differences
systemfit	Simultaneous Equation Estimation Package
tcltk	Interface to Tcl/Tk
tensor	Tensor product of arrays
tframe	Time Frame coding kernel
tkrplot	TK Rplot
tools	Tools for Package Development and Administration
tree	Classification and regression trees
tripack	Triangulation of irregularly spaced data
ts	Time series functions
tseries	Time series analysis and computational finance

```
twostage        Optimal design of two-stage-studies using the
                Mean Score method
vegan           Community Ecology Package
VLMC            VLMC -- Variable Length Markov Chains
waveslim        Basic wavelet routines for time series and
                image analysis
wavethresh      Software to perform wavelet statistics and
                transforms.
wle             Weighted Likelihood Estimation
xgobi           Interface to the XGobi and XGvis programs for
                graphical data analysis
XML             Tools for parsing and generating XML within R
                and S-Plus.
xtable          Export tables to LaTeX or HTML
```

Ab R 1.4.1 gibt es auch noch die Möglichkeit sich die vorhandenen Pakete per .packages(all. available =TRUE) anzusehen. Die Pakete lassen sich mit der Funktion library () bzw. require () laden, wobei require () zusätzlich mitteilt, ob der Ladevorgang erfolgreich war (innerhalb von Funktionen!). Ab R 2.5.0 schon über 1000 Pakete...

L.1.4 Datenimport

(a)
- **read.table()**
  davon mit speziellen Optionen abgeleitet:
  read.csv(), **read.csv2()**, **read**.delim(), **read**.delim2(),
  read.fwf()
- **scan()**

(b)
```
##################################################
# Loesungsvorschlag zu Blatt 1 Aufgabe 4(c)
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
```


```
####

# Einlesen des Datensatzes
dollar <- read.table(
              file = "dollar.data",
              header = TRUE,
              colClasses = c("character", "numeric")
              )

# Handelt es sich um einen data.frame?
is.data.frame(dollar)

# Spaltennamen
names(dollar)

# Zeilennamen
row.names(dollar)
```


L.1.5 Mustererzeugung

```
##################################################
# Loesungsvorschlag zu Blatt 1 Aufgabe 5
##################################################

# Zeile 1
M <- 1:20

# Zeile 2
M <- rbind(M, seq(from = 0.25, to = 5, by = 0.25))
# oder auch
M <- rbind(M, seq(from = 0.25, to = 5, length = 20))

# Zeile 3
M <- rbind(M, rep(1:2, rep(10,2)))
```


```
# Zeile 4
M ← rbind(M, rep(rep(1:2, rep(2,2)), 5))
# oder auch
M ← rbind(M, rep(1:2, rep(2,2)))
# erzeugt nur: 1 1 2 2
# jedoch: wird die Zeile automatisch gefuellt,
# indem dieser Vektor wiederholt wird

# Zeile 5
M ← rbind(M, rep(1:6, c(2, 2, 2, 4, 4, 6)))

# Zeile 6
M ← rbind(M, rep(rep(1:3, c(1,3,6)), 2))
```


L.2 Lösungsvorschläge Blatt 2

L.2.1 Indexoperationen, Matrizen

```
##################################################
# Loesungsvorschlag zu Blatt 2 Aufgabe 1
##################################################

##################################################
#Teil (a)
##################################################

# Matrix 1)
M1 <- diag(2, nrow=6)
M1[ col(M1) > row(M1) ] <- 1
# oder auch
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
M1 <- M1 + upper.tri(M1)

# Matrix 2
M2 <- diag(4, nrow=6)
M2[ (col(M2) - row(M2)) == 1] <- c(1,2,1,2,1)
M2[1,1] <- 2
M2[6,6] <- 2
# oder auch
M2 <- diag(c(2, rep(4,4), 2), nrow = 6)
M2[(col(M2) - row(M2)) == 1] <- c(1,2,1,2,1)

# Matrix 3
M3 <- matrix(rep(c(2:7), rep(6,6)),
             nrow = 6, ncol = 6)
M3 <- M3^(row(M3)-1)

##################################################
```


```
#Teil (b)
##########################################################

library(MASS)
data(painters)
namen <- row.names(painters)
namen[grep(pattern = 'e.*e.*e', namen)]
```


Peter Ruckdeschel
Matthias Kohl

L.2.3 Faktoren

```
##################################################
# Loesungsvorschlag zu Blatt 2 Aufgabe 3
##################################################

##################################################
#Teil (a)
##################################################

data(iris)
attach(iris)
tapply(Sepal.Length, Species, min)
tapply(Sepal.Width, Species, min)
tapply(Sepal.Length, Species, max)
tapply(Sepal.Width, Species, max)
```


Peter Ruckdeschel
Matthias Kohl

```
##############################################################
#Teil (b)
##############################################################
len <- as.matrix(tapply(
        Sepal.Length, Species, table)$setosa)
len[len == max(len)]
# bzw.
len[which.max(len)]
# gibt jedoch nur eines der beiden Maxima aus

wid <- as.matrix(tapply(
        Sepal.Width, Species, table)$setosa)
wid[wid == max(wid)]
# bzw.
wid[which.max(wid)]
# hier o.k., da es nur ein Maximum gibt
```


L.2.4 Einladen und Umformatieren von Datensätzen aus dem Netz

```
##################################################
# Loesungsvorschlag zu Blatt 2 Aufgabe 4
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

Filename <- "BL02Aufg4.txt"

#### Vorschlag basiert auf Loesung von Bianca Valentin
```


```
kursevonyahoo<-function(kurs,d1,m1,y1,d2,m2,y2,
                        zeitraum=365, tmpfile="temp.csv") {
    teil <- 0
    m1 <- m1 - 1  #yahoo will die Monatszahl-1
    m2 <- m2 - 1
    while(zeitraum > 0) {
        # URL aus den Einzelteilen zusammenbauen,
        # um csv-Datei herunterzuladen
        # dabei werden nur jeweils 200 Datensaetze
        # uebertragen, deswegen herunterladen in
        # Teilen à 200 Datensaetzen
        # mit "&y=teil" wird die Nummer des ersten
        # herunterzuladenden Datensatzes angegeben
        #
        url.teil1 <- "http://ichart.yahoo.com/table.csv?s="
        url<-paste(url.teil1, kurs, "&a=", m1, "&b=" ,d1,
```


```
                "&c=", y1, "&d=", m2, "&e=", d2, "&f=",
                 y2, "&y=", teil, "&g=d&ignore=.csv",
                 collapse="")

    download.file(url,tmpfile)

    if(teil == 0) {
        a <- read.csv(tmpfile, sep = ",")
    }
    else {
        tmp <- read.csv(tmpfile, sep=",")
        a <- rbind(a,tmp) # zwei Data.frames aneinanderfuegen
    }
    file.remove(tmpfile)
    teil <- teil + 200;
    zeitraum <- zeitraum - 200;
}
```


```
    n<-(dim(a))[1];
    row.names(a) <- 1:n
    #Aufraeumarbeiten -> Zeilennummern richtig setzen
    return(a);
}

kurs1 <- "BMW.DE"
kurs2 <- "DCX.DE"
bmw <-kursevonyahoo(kurs1,8,5,2006,
            8,5,2007)[,c("Date","Close")]
daimler<-kursevonyahoo(kurs2,8,5,2006,
            8,5,2007)[,c("Date","Close")]
```


L.2.5 String-, Matrixoperationen

```
##################################################
# Loesungsvorschlag zu Blatt 2 Aufgabe 5
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

Filename <- "BL02Aufg5.txt"

# Matrix 3 aus Aufgabe 1 (a)
M3 <- matrix(rep(c(2:7), rep(6,6)), nrow = 6,
```


```
                ncol = 6)
M3 <- M3^(row(M3)-1)

L <- list(Hilbert = M3)
EWs <- eigen(L$Hilbert)$values

# 1. Moeglichkeit
cat(paste("Die Eigenwerte der Matrix",
    names(L), "sind "), file = Filename)
cat(format(EWs, nsmall = 2, digits = 3),
    sep=", ", file = Filename, append = T)

# 2. Moeglichkeit
cat(paste("Die Eigenwerte der Matrix",
    names(L), "sind "), file = Filename)
cat(round(EWs, 2), sep = ", ",
    file = Filename, append=T)
```


L.3 Lösungsvorschläge Blatt 3

L.3.1 Umgang mit Zeichenketten

```
##############################################################
# Loesungsvorschlag zu Blatt 3 Aufgabe 1
##############################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

?grep
```


```
?regexp

### Einladen der Bibel
###       (bibschl.txt liege bereits entpackt im
###        Arbeitsverzeichnis)

a ← paste(scan("bibschl.txt",
                  what = character(0)), sep = "␣")
###
## Kontrolle
str(a)
a[1:10]

### (a)
### Laenge von a:
length(a)  ##747812
```


Peter Ruckdeschel
Matthias Kohl

```
### (b)
a1.0 ← gsub("[[:punct:]]", "", a)
a1.1 ← gsub("[[:cntrl:]]", "", a1.0)



### (c)
length(grep("Jesus|Jesu|Jesum|Jesse", a1.1)) ### 1007
length(grep("Menschensohn", a1.1)) ### 106

### (d)
a1 ← unique(a1.1)
length(a1) ##29584

### (e)
from ← c("ä", "Ä", "ö", "Ö", "ü", "Ü", "ß")
to ← c("ae", "Ae", "oe", "Oe", "ue", "Ue", "ss")
```


```
### mit for-Schleife
l ← length(from)
a.ohneUml ← a1.1

for (i in 1:l)
  {print(paste("Ersetze␣", from[i], "durch", to[i]))
   print(length(grep(from[i], a1.1)))
   a.ohneUml ← gsub(from[i], to[i], a.ohneUml)
  }


### (f)

Ind.jj ← grep("j.*j", a1, ignore.case = TRUE)
(a2.jj ← a1[Ind.jj])
```


```
#### (g)

Ind.je1 ← grep("j", a1, ignore.case = TRUE)
a.je1 ← a1[Ind.je1]

Ind.je2 ← grep("e", a.je1, ignore.case = TRUE)
(a.je ← a.je1[-Ind.je2])


### (h)
Ind.5 ← grep(".+{5}", a1)
a.5 ← a1[Ind.5]

Ind.5u ← grep("u", a.5, ignore.case = TRUE)
a.5u ← a.5[Ind.5u]

Ind.5uj ← grep("j", a.5u, ignore.case = TRUE)
```


```
a.5uj ← a.5u[Ind.5uj]
Ind.5ujJ ← grep("J", a.5uj, ignore.case = FALSE)
(a.5ujJ ← a.5uj[-Ind.5ujJ])
```


L.3.2 Umgang mit Zeichenketten II

```
##################################################
# Loesungsvorschlag zu Blatt 3 Aufgabe 2
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

### Einladen der Bibel
###       (bibschl.txt liege bereits entpackt im
###        Arbeitsverzeichnis)
```


```
a ← paste(scan("bibschl.txt",
                what = character(0)), sep = "␣")

### (a)
?LETTERS
LETTERS
sapply(LETTERS, function(i)
                length(grep(i, a, ignore.case = TRUE)))



### (b)
a1u ← unique(tolower(a1))
L1 ← sapply(a1u, nchar)
(max(L1))

O ← order(L1, decreasing = TRUE)
a1uo ← a1u[O]
```


Peter Ruckdeschel
Matthias Kohl

```
a1uo[1:10]

### (c)
Zahl1 <- "eins|tausend|hundert|zwei"
Zahl2 <- "|drei|vier|fünf|sechs|sieben"
Zahl3 <- "|acht|neun"
Zahl <- paste(Zahl1,Zahl2,Zahl3,sep = "")
IndZahl <- grep(Zahl,a1uo)
(a1uo[-IndZahl])[1:10]

### (d)
a1l <- tolower(a1.1)
L2 <- sapply(a1l, nchar)
table(L2)

### (e)
```


```
Ta ← table(a1l)
Tas ← sort(Ta, decreasing = TRUE)
Tas[1:10]
```


L.3.3 Umgang mit regulären Ausdrücken / regexp's

```
#####################################################
# Loesungsvorschlag zu Blatt 3 Aufgabe 3
#####################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

## Zusammenbauen der URL

URL1 <- "http://www.uni-bayreuth.de/departments/math"
```


```
URL2 ← "/org/mathe7/services.html"
URL ← paste(URL1,URL2,sep="")

download.file(URL, destfile = "test.html")
file.show("test.html")

### alles in einen langen String schreiben
a ← paste(scan("test.html", what = character(0)),
          sep="␣", collapse="␣")
a
### in einzelne < ... > Tags zerlegen
a2 ← strsplit(gsub(">", ">>\n", a),
              ">\n")[[1]]
a2
length(a2)

a3 ← a2[[1]]
```


```
a4 <- grep("<.*a .*href.*>", a3,
           ignore.case = TRUE, value = TRUE)

### <a href - Tags heraussuchen und
##    inneren Teil rausschreiben
gsub(".*<a href=\"(.*)\">","\\1", a4)
```


Peter Ruckdeschel
Matthias Kohl

L.3.4 Matrixoperationen

```
##################################################
# Loesungsvorschlag zu Blatt 3 Aufgabe 4
##################################################

# Teil (a)
M <- matrix(0, nrow=6, ncol=6)
M <- 1/(row(M)+col(M)-1)

#Teil (b)
det(M)
#Kondition
val <- eigen(M, symmetric = TRUE)$values
abs(max(val))/abs(min(val))
##oder
kappa(M, exact = TRUE)
```


```
#Teil (c)
b <- 1:6
solve(M, b)
#oder:
qr.solve(M, b)
#evtl wegen der schlechten Kondition
```


L.3.5 Schreiben von Daten auf File

```
##################################################
# Loesungsvorschlag zu Blatt 3 Aufgabe 5
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

Filename ← "erg.txt"

# verwenden Matrix 3 aus Aufgabe 1 (a)
M3 ← matrix(rep(c(2:7), rep(6,6)), nrow=6, ncol=6)
```


```
M3 ← M3^(row(M3)-1)

# Ausgabe
cat("Die Matrix M lautet \n",
      sep = c("",rep(c(rep(" , " ,5),"\n"),6)),
      format(t(M3), digits = 5, nsmall = 3),
      file = Filename)
cat("Die Eigenwerte der Matrix M sind \n(",
      sep = c("", rep(" , " ,5), "\n"),
      format(eigen(M3)$values, digits = 5,
             nsmall = 3),
      ")", file = Filename,
      append = TRUE)
```


Peter Ruckdeschel
Matthias Kohl

L.4 Lösungsvorschläge Blatt 4

L.4.1 Skalenniveaus

- Geschlecht (kategorielles Merkmal): Modus
- Schulnoten, Semesteranzahl (ordinale Merkmale): Modus, Median, mit Einschraenkungen ist auch Mittelwert bzw. Note 2.3 sinnvoll (ist die 1 von der 2 so "weit"weg wie die 3 von der 4)
- Ideale Normalverteilung (metrisches Merkmal): Mittelwert, Median, Modus nicht sinnvoll — es sei denn als Modus der Dichte!!!

L.4.3 Univariate Analyse

```
##################################################
# Loesungsvorschlag zu Blatt 4 Aufgabe 3
##################################################

##################################################
# Teil(a)
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####
# Download von kredit1.txt aus Netz in Datei
```


```
# kredit1.txt im Arbeitsverzeichnis

kredit <- read.table(file="kredit1.txt",
                     header = TRUE)
attach(kredit)
names(kredit)

stem(Laufzeit)
summary(Laufzeit)
stem(Kredithoehe)
summary(Kredithoehe)
table(Zahlungsmoral)

##############################################################
# Teil(b)
##############################################################
```


```
par(mfrow=c(2,2))
boxplot(Laufzeit)
title("Boxplot der Laufzeit")
hist(Laufzeit)
boxplot(Kredithoehe)
title("Boxplot der Kredithoehe")
hist(Kredithoehe)

detach()
```


L.4.4 Elementare Datenanalyse

```
##################################################
# Loesungsvorschlag zu Blatt 4 Aufgabe 4
##################################################

##################################################
# Teil(a)
##################################################
# Vorarbeit:
## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####
# Download von baby.txt aus Netz in Datei
```


```
# baby.txt im Arbeitsverzeichnis

a <- read.table(
        file = "baby.txt", sep = ",",
        header = FALSE, skip = 2)
# Ueberlesen der ersten beiden Zeilen (skip=..)
# Trennungszeichen ist ","

# Umwandeln in eine Matrix
a <- data.matrix(a)
M <- a[,1]  # wie in Angabe Masse bei Geburt
R <- a[,2]  # wie in Angabe rel. Massezuwachs
          # in der ersten Woche

##################################################
# Teil(b)
##################################################
```


```
# Histogramm von R
hist(R)



##################################################
# Teil(c)
##################################################
# Berechnen einiger Kenngroessen
# der empirischen Verteilung
mean(M)
median(M)
var(M)
sd(M)
cor(cbind(M,R))

max(M) # nicht Modalwert!
# Modalwert
```


```
modal <- as.matrix(table(M))
modal <- modal[modal == max(modal)]
# bzw.
modal <- modal[which.max(modal)]


##################################################
# Teil(d)
##################################################
boxplot(R)
```


L.4.5 Elementare Datenanalyse II

```
#####################################################
# Loesungsvorschlag zu Blatt 4 Aufgabe 5
#####################################################

#####################################################
# Vorarbeit:
## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####
### aus Blatt 2 Aufgabe 4:

#kursevonyahoo<- function(kurs,d1,m1,y1,d2,m2,y2,
```


```
#                   zeitraum=365, tmpfile="temp.csv")
#{ .... }

##################################################
# Teil(a)
##################################################
kurs1 <- "BMW.DE"
kurs2 <- "DCX.DE"
bmw <- kursevonyahoo(kurs1,22,5,2005,
          22,5,2006)[,c("Date","Close")]
daimler <- kursevonyahoo(kurs2,22,5,2006,
          22,5,2006)[,c("Date","Close")]


##################################################
# Teil(b)
##################################################
```


```
DF ← data.frame(cbind(daimler, bmw)),
names(DF)=c("Daimler","BMW")

##################################################
# Teil(c)
##################################################
n←nrow(DF)
DF0 ← DF[2:n,]/DF[1:(n-1), ]-1
### oder
## DF0 ← apply(DF,2, diff)-1

boxplot(DF0)

##################################################
# Teil(d)
##################################################
cor(DF0)
```


L.5 Lösungsvorschläge Blatt 5

L.5.1 Univariate Konvexkombinationen

```
##################################################
# Loesungsvorschlag zu Blatt 5 Aufgabe 1
##################################################

##################################################
# Teil (a)
##################################################
n <- 100
X <- rnorm(n, 0 + 3*rbinom(n,1,0.1), 1)

##################################################
# Teil (b)
```


```
#############################################################
stem(X)
summary(X)

#############################################################
# Teil (c)
#############################################################
boxplot(X)

#############################################################
### einige Erlaueterungen
#############################################################
#Teil (a)
### didaktisch klarer:
#
Xid <- rnorm(n, mean = 0, sd = 1)
    # geht auch: rnorm(n)
```


```
Xc <-  rnorm(n, mean = 3, sd = 1)
    # geht auch: rnorm(n,mean=3)
### Auswahlmechanismus, der steuert ob
# ideal oder kontaminiert
# => Variable U = 0 mit Ws 0.9
#             und = 1 mit Ws 0.1
# also L(U) = Bin(1,0.1)
U <- rbinom(n, size = 1, prob = 0.1)
Xre <- (1-U)*Xid+U*Xc

### Verteilung von Xre:
#  P(Xre<=t) =
#  = P((1-U)*Xid+U*Xc<=t) =
#  = P((1-U)*Xid+U*Xc<=t, U=1) +
#    P((1-U)*Xid+U*Xc<=t, U=0) =
#  = P(Xc<=t, U=1) +
#    P(Xid<=t, U=0) =
```


```
#         # {[U,Xid] sto.u., [U,Xc] sto.u.}
#  = P(Xc<=t)*P(U=1) +
#    P(Xid<=t)*P(U=0) =
#  = 0.1*P(Xc<=t)+0.9* P(Xid<=t)
#  = 0.1*pnorm(t,mean=3,sd=1) +
#    0.9*pnorm(t,mean=0,sd=1)
#########################################
## ACHTUNG: das ist nicht die Verteilung von
X0 <- 0.9*Xid+0.1*Xc
##
#spaeter in Stochastik I:
#   P(X0<=t)=pnorm(t,mean=0.9*0+0.1*3,
#       sd=sqrt(0.9^2*1^2+0.1^2*1^2))
####
#Unterschied: (nicht verlangt!)
##
x <- seq(-6,6,by=0.01)
```


```
# mit Dichten
did ← dnorm(x, mean = 0, sd = 1)
dc ←  dnorm(x, mean = 3, sd = 1)

d ← 0.9*did+0.1*dc

d0 ← dnorm(x, mean = 0.3,
           sd = sqrt(0.82))
matplot(x, cbind(did,d,d0),
        lwd = 4, type = "l")
##
#schwarz: Dichte von Xid
# gruen: Dichte von X0
# rot: Dichte von X
##
# mit Verteilungsfkt.
pid ← pnorm(x, mean = 0, sd = 1)
```


```
pc  <- pnorm(x, mean = 3, sd = 1)
p<- 0.9*pid + 0.1*pc
p0 <- pnorm(x, mean = 0.3,
            sd = sqrt(0.82))
matplot(x, cbind(pid,p,p0),
        lwd = 4, type = "l")
##
#schwarz: Vtlgsfkt von Xid
# gruen: Vtlgsfkt von X0
# rot: Vtlgsfkt von X
##

#Teil (b)
stem(X)
summary(X)
# vgl:
stem(Xre)
```


```
summary(Xre)
stem(Xid)
summary(Xid)

###Interpretaion:
# Mittelwert,Maximum von X und Xre wird
# ausgelenkt gegenueber Xid

#Teil (c)
boxplot(list(X = X, Xre = Xre, Xid = Xid))

## Interpretation:
# generell Auslenkung nach rechts von
# X und Xre gegenueber Xid
### Unterschied X, Xre kommt nur durch
# "Zufall" (2 verschiedene Stichproben
# aus derselben Verteilung
```


L.5.4 Umgang mit Zusatzpaketen distr / distrEx

```
##################################################
# Loesungsvorschlag zu Blatt 5 Aufgabe 4
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####


##a##
raus<-function(x){c("E" = E(x),
```


```
                    "sd" = sd(x),
                    "med" = median(x))}

require("distrEx")
X1 ← Pois(0.3)
X2 ← Td(df = 7, ncp = 0)
X3 ← log(1+exp(X1+3.5*X2))

lapply(list(X1,X2,X3),raus)

##b##

CLTplot←function(x,n){
i←0; Sn←0
while(i<n)
      {i←i+1; Sn ← Sn+x}
# Sn0 ← (Sn/n-E(x))/sd(x)*sqrt(n)
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
##besser:
Sn0 ← (Sn-E(Sn))/sd(Sn)
x ← seq(-4,4,0.01)
Sx ← d(Sn0)(x)
dx ← dnorm(x)
matplot(col = c("green","red"), x, cbind(Sx,dx),
        title = paste("Summe der Ordnung n=",
                      n, " von ", x),
        type = "l", lwd = 2,
        ylab = paste("n=",n))
}

###nicht verlangt
tues← function(x){
par(mfrow = c(3,4))
for(i in 1:10) CLTplot(x,n=i)
CLTplot(x, n = 20)
```


```
CLTplot(x, n = 50)
par(mfrow = c(1,1))}

tues(x = Exp(1))
tues(x = sin(Exp(1)))

##################################################
```


L.5.5 Übungsaufgaben zur Stochastik

```
##################################################
# Loesungsvorschlag zu Blatt 5 Aufgabe 5
##################################################

##################################################
# Teil (a)
##################################################
# Wahrscheinlichkeit, dass wenigstens n
# am gleichen Tag Geburtstag haben fuer
# n=1,...,365
Pcum <- 1-cumprod(365-1:365)/365^(1:365)
Pcum[is.na(Pcum)] <- 1
# aufgrund numerischer Schwierigkeiten muss
# Pcum irgendwann von Hand 1 auf gesetzt
# werden!
```


```
# einfacher:
PP0<-cumsum(log(365-1:365))-(1:365)*log(365)
Pcum <- 1-exp(PP0)
# alle Ergebnisse die kleiner als 0.25 sind
# und einer dazu
ergebnis <- sum(Pcum<0.25) + 1
ergebnis


##########################################################
#Teil (b)
##########################################################
# Bezeichnungstabelle
# Angabe Blatt          R
#        N              n+M
#        n              m
#        m              k
#
```


```
#          k                    q

pte ← 1-phyper(q=23, m=120, n=1:1000, k=100)
ergebnis ← sum(pte<0.05) + 1 + 120
ergebnis

# also mit Wahrscheinlichkeit 95% ist N<560
# Kontrolle: untere Schranke
pte ← phyper(q=23, m=120, n=1:1000, k=100)
ergebnis ← sum(pte<0.05) + 1 + 120
ergebnis



##################################################
#Teil (c)
##################################################
# Zahl der Paare
```


```
N ← 20
einzelSocke ← 0:N
SockeninTrommel ← 2*N − einzelSocke
PaarSockeninTrommel ← SockeninTrommel −
                      einzelSocke

# im Schritt k kann entweder ein bereits
# aussen liegender Socke gezogen werden und
# es gibt ein Paar -> Ppaarcond
# oder nicht -> Pnpaarcond
Ppaarcond ← einzelSocke/SockeninTrommel
Pnpaarcond ← 1−Ppaarcond

# wieder logTrafo zum stabileren
# Rechnen mit Fakultaeten
lPnpaarcond ← log(Pnpaarcond)
```


```
# Wahrscheinlichkeit bis Zug k kein
# Paar bekommen zu haben
Pncum ← exp(cumsum(lPnpaarcond))
# verschieben um Wahrscheinlichkeit
# dass in Zug k erstes Paar
Ppaarcond ← c(Ppaarcond,0)
Pncum ← c(1,Pncum)
pte ← Pncum*Ppaarcond

# Wahrscheinlichkeiten
# pte
sum(pte) # Kontrolle

# Wahrscheinlichkeit bis zum k ten Zug
# ein Paar = 1 - Pncum
# cumsum(pte)
sum(cumsum(pte)<0.7) + 1
```


L.6 Lösungsvorschläge Blatt 6

L.6.1 Visualisierung des (schwachen) Gesetzes der großen Zahlen

```
##################################################
# Loesungsvorschlag zu Blatt 6 Aufgabe 1
##################################################

# wir schreiben das ganze gleich
# als Funktion in N und M

LLN <- function(N,M){
##################################################
# Teil (a)
##################################################
```


Peter Ruckdeschel
Matthias Kohl

```
# Ziehen der Wuerfelergebnisse
# ---- (M*N Stueck)
    W <- sample(seq(1:6), size = N*M,
                replace = TRUE)
# daraus Ableiten des Indikators
# "Habe eine 6"
    s <- (W == 6)
# in Matrix -> M^="runs" N^=Stichproben
    s <- matrix(s, M, N)

##################################################
# Teil (b)
##################################################
# fuer jede der M Stichproben:
# Berechnen des Stichprobenmittelwertes
    S <- apply(s, 1, mean)
```


```
##################################################
# Teil (c)
##################################################
    hist(S)
    boxplot(S)

##################################################
#Teil (d)
##################################################
    VN <- mean((S-1/6)^2)

    return(list(VN, S))
}


##################################################
# Ausfuehren der Funktion
```


```
##########################################################
# verschiedene n-Werte
n <- c(1,3,5,10,50,100,1000)

# ohne for-Schleife!
erg <- sapply(n, LLN, M=10000)
VNsim <- matrix(unlist(erg[1,]), ncol = 1)

SS <- data.frame(matrix(unlist(erg[2,]),
        nrow = 10000, ncol = length(n)))
names(SS) <- paste(n)

boxplot(SS)
dev.off()
## suggeriert abklingen mit 1/N
#  Vorfaktor erraten: 5/36
#  Vergleich Theorie / Simulation
```


```
VNtheo ← 1/36/n
VNa ← cbind(VNsim,VNtheo)
matplot(n, VNa, type = "l")

par(new = TRUE)
par(mfrow = c(3,2))
myhist←function(iS,XX,siz,...){
    hist(XX[,iS],
         main = paste("n=", siz[iS]), ...)}
sapply(1:7, myhist, XX = SS, siz = n,
    xlab = "S_i=rel. Hauefigkeit einer 6",
    ylab = "# S_i=.. in 10000 runs")
```


L.6.3 Numerische Integration: Berechnung von π

```
##################################################
# Loesungsvorschlag zu Blatt 6 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################
n <- 1000
U <- matrix(runif(n = 2*n, min = 0, max = 1),
            nrow = n, ncol = 2)

##################################################
# Teil (b)
##################################################
K <- ((U[,1]-0.5)^2 + (U[,2]-0.5)^2 <= 0.25)
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
pi4 ← sum(K)/n
pi.emp ← 4*pi4

##################################################
# Teil (c)
##################################################
# Normalapproximation :
#       sqrt(n) (p_n-p) -> N(0,p(1-p))
# mit p_n relativer Anteilswert;
# konservativ: p(1-p)<=1/4

# P( |p_n-p|> 10^-6)=
#       P( sqrt(n)|p_n-p|> sqrt(n) 10^-6)
#     ~ P(|N(0,1/4)|> sqrt(n) 10^-6)  !<= 0.95
# <=>  2 Phi(2 sqrt(n) 10^-6)-1 >= 0.95
# <=>  Phi(2 sqrt(n) 10^-6) >= 0.975
# <=>  2 sqrt(n) 10^-6 >= Phi^-1(0.975)
```


```
# <=>  sqrt(n) >= 2Phi^-1(0.975) 10^6
# <=>  n >= 4 x 10^12

# Zusaetzlich: (Erlaeuterung spaeter)
# einplotten in einen Kreis

##################################################
# Berechnung und Plot des Kreises
##################################################
x <- seq(from = 0, to = 1, by = 0.001)
y1 <- 0.5 + sqrt(x-x^2)
y2 <- 0.5 - sqrt(x-x^2)
li <- -0.2
re <- 1.3
par(col = 1)
plot(x, y1, type = "l", xlim = c(li,re),
     ylim = c(li,re), xlab = "x-Achse",
```


```
     ylab = "y-Achse", lwd = 2)
lines(x, y2, lwd = 2)

# Plot des Quadrates
lines(c(0,1,1,0,0), c(0,0,1,1,0), lwd = 2)

# Plot der Punkte
im.Kreis <- U*K
im.Kreis <- matrix(im.Kreis[im.Kreis != 0],
                   nrow = sum(K), ncol = 2)
aus.Kreis <- U*(1-K)
aus.Kreis <- matrix(aus.Kreis[aus.Kreis != 0],
                   nrow = sum(1-K), ncol = 2)
par(col = 2, pch = 1)
points(im.Kreis[,1], im.Kreis[,2])
par(col = 3, pch = 4)
points(aus.Kreis[,1], aus.Kreis[,2])
```


```
par(col = 1)
title(paste("Schaetzung der Kreiszahl pi =",
      round(pi, 6)))
legend(li, re, legend = c("im Kreis", "ausserhalb"),
       pch = c(1,4), col = c(2,3))
text(0.5, -0.1, labels = paste("Der aktuelle",
    "Schaetzwert fuer pi ist", pi.emp))
```


L.6.4 Berechnung von $\mathbf{E}[\chi_1^2]$

```
##################################################
# Loesungsvorschlag zu Blatt 6 Aufgabe 4
##################################################

##################################################
# Teil (a)
##################################################
# X~N(0,1)
# E[Y]=E[X^2]=1



##################################################
# Teil (b)
##################################################
M ← 10000
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
n <- 100
x <- matrix(rchisq(M*n, df = 1), nrow = M,
            ncol = n)
mws <- apply(x, 1, mean)
varg <- var(mws)
print(c(mean(mws),varg))
# analytisch:
#   Var[ave(Y_i)]     =
# = Var[ave(X_i^2)]   =
# = (E[X^4]-E[X^2]^2)/n = 2/n



##################################################
# Teil (c)
##################################################
#Abschneidepunkt wo
#       int_c^Inf x f_x dx < 10^-6
```


```
# Variablentrafo ergibt
#       int_c'^Inf gamma(a=3/2)[x] dx < 10^-6
# Abschätzung des Integrals durch
#            gamma(a=5/2)[x]
T <- qgamma(1-10^-6, shape = 5/2)
U <- matrix(runif(M*n, min = 0, max = T),
            nrow = M, ncol = n)
fs <- dchisq(df = 1, x = U)*U
mws2 <- T*apply(fs, 1, mean)
varg2 <- var(mws2)
print(c(mean(mws2),varg2))



##################################################
# Teil (d)
##################################################
U21 <- U[,1:(n/2)]
```


```
U22 <- T - U21
fs1 <- dchisq(df = 1, x = U21)*U21
fs2 <- dchisq(df = 1, x = U22)*U22
mws3 <- T*apply(cbind(fs1, fs2), 1, mean)
varg3 <- var(mws3)
print(c(mean(mws3),varg3))


##################################################
# Teil (e)
##################################################
x <- matrix(rnorm(M*n), nrow = M, ncol = n)
mws4 <- apply(x^2, 1, mean)
varg4 <- var(mws4)
print(c(mean(mws4),varg4))

##################################################
```


```
# Erweiterung: antithetisch mit chisq
##########################################################
V <- matrix(runif(M*n), nrow = M, ncol = n)
V21 <- V[,1:(n/2)]
V22 <- 1 - V21
x21 <- qchisq(p = V21, df = 1)
x22 <- qchisq(p = V22, df = 1)
mws6 <- apply(cbind(x21, x22), 1, mean)
varg6 <- var(mws6)
print(c(mean(mws6),varg6))
```


L.6.5 Konfidenzintervalle, Bootstrap

```
##################################################
# Loesungsvorschlag zu Blatt 6 Aufgabe 5
##################################################

n <- 21

##################################################
# Teil (a)
##################################################
stichpa <- rnorm(n)
meda <- median(stichpa)
# Berechnung der Bandweiten mit
# verschiedenen Methoden
d1 <- density(stichpa, n = 1, from = meda,
      to = (meda+0.01), bw = bw.nrd0(stichpa))$y
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
d2 ← density(stichpa, n = 1, from = meda,
      to = (meda+0.01), bw = bw.nrd(stichpa))$y
#d1 ← density(stichpa, n = 1, from = meda,
#      to = (meda+0.01), bw = bw.ucv(stichpa))$y
#d2 ← density(stichpa, n = 1, from = meda,
#      to = (meda0.01), bw = bw.bcv(stichpa))$y
streua ← 1/(2*sqrt(n)*(d1+d2)/2)
Kla ← qnorm(c(0.025,0.975))*streua


#########################################################
# Teil(b)
#########################################################
N ← 10000
stichpb ← matrix(rnorm(n*N), nrow = N, ncol = n)
medb ← apply(stichpb, 1, median)
medb.emp ← mean(medb)
```


```
streub ← sd(medb)
KIb ← quantile(medb, c(0.025, 0.975))


##################################################
# Teil (c)
##################################################
M ← 1000
stichpc ← matrix(sample(stichpa, size = n*M,
                        replace = TRUE), nrow = M,
                        ncol = n)
medc ← apply(stichpc, 1, median)
medc.bs ← mean(medc)
streuc ← sd(medc)
KIc ← quantile(medc, c(0.025, 0.975))

##################################################
```


```
# Erweiterung
##################################################
m <- 25
medf.bs <- numeric(m)
streuf <- numeric(m)
KIf <- matrix(0, nrow = m, ncol = 2)

for(i in 1:m)
{
    stichpf1 <- rnorm(n)
    stichpf <- matrix(sample(stichpf1, size = n*M,
                    replace = TRUE), nrow = M,
                    ncol = n)
    medf <- apply(stichpf, 1, median)
    medf.bs[i] <- mean(medf)
    streuf[i] <- sd(medf)
    KIf[i,] <- quantile(medf, c(0.025, 0.975))
```


```
}

medf.bsm ← mean(medf.bs)
streuf.bsm ← mean(streuf)
Klf.bsm ← apply(Klf, 2, mean)

##########################################################
# Teil(d): Vergleich der Ergebnisse
##########################################################
cat("\n==========================================================\n")
cat("Die Ergebnisse fuer den Median:\n")
cat("Teil(a):\t", meda, "\n")
cat("Teil(b):\t", medb.emp, "\n")
cat("Teil(c):\t", medc.bs, "\n")
cat("Teil(c) Erweiterung:\t", medf.bsm, "\n")
cat("==========================================================\n")
cat("Die Ergebnisse fuer die Streuung:\n")
```


```
cat("Teil(a):\t", streua, "\n")
cat("Teil(b):\t", streub, "\n")
cat("Teil(c):\t", streuc, "\n")
cat("Teil(c) Erweiterung:\t", streuf.bsm, "\n")
cat("==========\n")
cat("Die Konfidenzintervalle:\n")
cat("Teil(a):\t", KIa, "\n")
cat("Teil(b):\t", KIb, "\n")
cat("Teil(c):\t", KIc, "\n")
cat("Teil(c) Erweiterung:\t", KIf.bsm, "\n")
cat("==========\n")
```


L.7 Lösungsvorschläge Blatt 7

L.7.1 Maximale Lücke

```
#############################################################
# Loesungsvorschlag zu Blatt 7 Aufgabe 1
#############################################################

#############################################################
# zuerst mit "for"-Schleife
#############################################################

# Funktion zur Berechnung der Minima
# U1 ist ein Array der Dimension n x k x M
minima <- function(U1){
    M <- dim(U1)[3]
```


```
n ← dim(U1)[1]
minima ← vector("numeric", length = M)

for(i in 1:M){
    U2 ← U1[,,i]
    erg ← Inf
    for(j in 1:n)
        for(k in 1:n){
            if( k != j ){
                vek ← U2[j,]-U2[k,]
                erg1 ← t(vek) %*% vek
            }
            else{
                erg1 ← Inf
            }

            if(erg > erg1) erg ← erg1
```


```
            }

        minima[i] <- erg
    }

    return(minima)
}


k <- 2
M <- 500

# n = 5
n <- 5
U1 <- matrix(runif(M*k*n,0,1), nrow = M*n, ncol = k)
U1 <- array(U1, dim = c(n,k,M))
Min1 <- minima(U1)
```


```
# n = 25
n ← 25
U1 ← matrix(runif(M*k*n,0,1), nrow = M*n, ncol = k)
U1 ← array(U1, dim = c(n,k,M))
Min2 ← minima(U1)

# n = 100
n ← 100
U1 ← matrix(runif(M*k*n,0,1), nrow = M*n, ncol = k)
U1 ← array(U1, dim = c(n,k,M))
Min3 ← minima(U1)

# n = 500
n ← 500
U1 ← matrix(runif(M*k*n,0,1), nrow = M*n, ncol = k)
U1 ← array(U1, dim = c(n,k,M))
```


```
Min4 ← minima(U1)

summary(Min1)
summary(Min2)
summary(Min3)
summary(Min4)

# gruppierter Boxplot
boxplot(list(Min1 = Min1, Min2 = Min2, Min3 = Min3,
             Min4 = Min4), main = "Boxplots")

# neues Graphikfenster
windows()
# 2 x 2 Bilder im Graphikfenster
par(mfrow = c(2,2))
hist(Min1, main = "n=5")
hist(Min2, main = "n=25")
```


```
hist(Min3, main = "n=100")
hist(Min4, main = "n=500")


##################################################
# vektorwertige Alternative - im Vergleich
# Idee nur 2 dimensionen -> hier "for"
# nicht schlimm
# aber viele Beobachtungen -> hier "for"
# schlimm
##################################################
# vektorwertige Formulierung
luecke.vec <- function(n) { #erst die x Koordinate
    X.x <- runif(n)

# erzeuge eine Matrix mit identischen Zeilen
    X.xx1 <- matrix(X.x, nrow = n, ncol = n)
```


```
# Matrtix mit identischen Spalten
    X.xx2 ← t(X.xx1)

# Matrix mit Eintraegen (X_i-X_j)^2_{i,j}
    X.xd ← (X.xx1-X.xx2)^2

# Diagonale aus Minimumbildung ausschliessen
    X.xd[row(X.xd) == col(X.xd)] ← 2

# Platz machen
    rm(X.xx1,X.xx2)

# das gleiche fuer die y Koordinate
    X.y ← runif(n)
    X.yy1 ← matrix(X.y, nrow = n, ncol = n)
    X.yy2 ← t(X.yy1)
    X.yd ← (X.yy1-X.yy2)^2
```


```
    X.yd[row(X.yd) == col(X.yd)] <- 2

# Platz machen
    rm(X.yy1,X.yy2)

# Addition der Koordinaten
    X.d <- X.xd + X.yd

    return(min(sqrt(X.d)))
}

# Berechnen von M Simulationsdruchgaengen
# ohne "for"
dluecke.vec <- function(X,n){luecke.vec(n)}
Mluecke <- function(M,n){
    sapply(1:M,dluecke.vec,n = n)}
```


```
# Berechnen von M Simulationsdruchgaengen
# ohne "for" fuer verschiedene n
Mnluecke <- function(X, M, fkt = min){
    fkt(Mluecke(M = M,n = X))}

M <- 100
n <- c(5, 25, 100, 500)

Minluecke <- function(M, n, afkt = min){
    sapply(n,Mnluecke,M = M,fkt = afkt)}

Minluecke(M = M, n = n, summary)
Minluecke(M = M, n = n, hist)
Minluecke(M = M, n = n, boxplot)
```


L.7.2 Buffons Nadelproblem — Berechnung von π II

```
##################################################
# Loesungsvorschlag zu Blatt 7 Aufgabe 2
##################################################

##################################################
# Hilfestellung fuer Buffons Nadel Problem
# Zeichnen der Streifen-Enden
##################################################

# Streifen-Abstand
d <- 2

##################################################
# damit das ganze nicht perspektivisch
```


Peter Ruckdeschel
Matthias Kohl

```
# verzerrt wirkt, soll das Fenster ein
# Quadrat, hier [-10,10]^2, abbilden
##################################################
# x-Gitterwerte
x <- seq(-10, 10, by = d)
m <- length(x)

# wir verwenden die Funktion "lines"
# diese verlangt jeweils ein Argument x und y:
#  --- jeweils eine Liste der x[y]-Koordinaten
#  der Linien der Form
#      x1-1.Endpunkt,x1-2.Endpunkt,NA (zum Trennen!),
#      x2-1.Endpunkt,x2-2.Endpunkt,NA (zum Trennen!),
#      ...
# (y entsprechend)

# die Y-Koordinaten der Streifen-Enden sind
```


```
# jeweils -10 und 10
y <- x*0-10  # die unteren Koordinaten

# jetzt Erzeugung einer Matrix xx der Form
# x1-1,x1-2,NA
# x2-2,x2-2,NA
# ....
# xm-1,xm-2,NA
#
# yy entsprechend
xx <- cbind(x, x, x*NA)
yy <- cbind(y, -y, y*NA)

# Umwandeln in die entsprechenden Listen
xx <- as.vector(t(xx))
yy <- as.vector(t(yy))
```


```
# Erzeugen des Koordinatenssystems
# (eigentlich Plot der Punkte (-10,-10) und (10,10)
# aber die beiden Punkte werden durch type="n"
# nicht geplottet; ebenfalls keine
# x- und y-Achsenbezeichnung)
plot(c(-10,10), c(-10,10), type = "n",
     xlab = "", ylab = "")
# nun die Linien
lines(x = xx, y = yy)



##################################################
# Teil (a): Ziehung der Nadeln
##################################################
l<-1
N<-10000
```


```
# Nadelmittelpunkte
xmid <- runif(N, min = -10, max = 10)
ymid <- runif(N, min = -10, max = 10)

# Winkel
angl<-runif(N,min = 0,max = 2*pi)

# Umrechnung in Endpunkte
cangl<-cos(angl)
sangl<-sin(angl)
xl<-xmid-l*cangl/2
xr<-xmid+l*cangl/2
yl<-ymid-l*sangl/2
yr<-ymid+l*sangl/2



##################################################
```


Peter Ruckdeschel
Matthias Kohl

```
# Teil (b)/(c): Schnitt von Nadel i mit
# einer Streifengrenze j, falls
# sign(xl[i]-M[i,j])<>sign(xr[i]-M[i,j])
##################################################
sxl <- sign(xl - t(matrix(x, length(x), N)))
sxr <- sign(xr - t(matrix(x, length(x), N)))
ind <- apply( (sxr*sxl == -1), 1, sum)

##################################################
# Teil (d)
##################################################
dl <- -1e-3
dr <- 1e-3
Pexakt <- function(n, pr = 1/pi, ob = 1/(pi+dl),
                   unt = 1/(pi+dr)){#
pbinom(n*ob, size = n,
    prob = pr) - pbinom(n*unt, size = n, prob = pr)
```


```
}

Pexakt(10^6)
Pexakt(10^7)
Pexakt(10^8)
Pexakt(.85*10^8)
Pexakt(.82*10^8)
Pexakt(.81*10^8)
Pexakt(.815*10^8)
Pexakt(.813*10^8)
Pexakt(.812*10^8)
Pexakt(.8125*10^8)
Pexakt(.8123*10^8)
Pexakt(.8121*10^8)
Pexakt(.8120*10^8)
Pexakt(.81205*10^8)
Pexakt(.81201*10^8)
```


```
Pexakt(81201000)
Pexakt(81200000)
# zu ungenau!
###

#########################################################
# Teil (e)
#########################################################
# Vorbereitung fuer lines
XX <- cbind(xl, xr, xr*NA)
YY<-cbind(yl, yr, yr*NA)
lines(list(x = as.vector(t(XX)), y = as.vector(t(YY))),
      lty = 1)
# Schnitt - Nadeln in rot
XXc <- XX[ind == 1,]
YYc <- YY[ind == 1,]
lines(list(x = as.vector(t(XXc)),y = as.vector(t(YYc))),
```


```
       lty = 1, col = "red")
1/mean(ind)

##################################################
# Teil (f): Methode Blatt 4 Aufgabe 2 viel
# besser (ca. Faktor 100 mal weniger
# Beobachtungen noetig fuer gleiche
# Genauigkeit)
##################################################
```


Peter Ruckdeschel
Matthias Kohl

L.7.3 Dichteplot

```
##################################################
# Loesungsvorschlag zu Blatt 7 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################
dichteplot ← function(vtlg, typ, ...){
    quart ← numeric(3)
    for(i in 1:3)
        eval(parse(text=paste("quart[i]←q", vtlg,
                "(i/4, ...)", sep="")))

    eval(parse(text=paste("links←q",
                vtlg,"(0.05, ...)", sep="")))
```


```
eval(parse(text=paste("rechts<-q",
                vtlg,"(0.95,...)", sep="")))

if(typ=="diskret"){
    x <- seq(from=links, to=rechts, by=1)
    eval(parse(text=
        paste("plot(x,d",vtlg,"(x,...),",
          "type=\"s\",ylab=\"WS-Funktion\")",
          sep="")))
    title(paste("Beispiel einer WS-Funktion",
          "fuer die Verteilungsklasse", vtlg))
}
else{
    x <- seq(from=links, to=rechts, by=0.01)
    eval(parse(text=
        paste("plot(x,d",vtlg,"(x,...),",
            "type=\"l\",ylab=\"Dichte\")",
```


```
                    sep="")))
        title(paste("Beispiel eines Dichteplots",
                    "fuer die Verteilungsklasse", vtlg))
    }

    return(as.list(quart))
}

#####################################################
#Teil (b)
#####################################################
# prompt(dichteplot)
# Erzeugt Datei dichteplot.Rd im aktuellen
# Verzeichnis
# Einfaches Editieren in einem Editor moeglich
# Anschliessend: Kopieren der Datei in das
# Verzeichnis "man"
```


```
# der library, zu der die Funktion gehoert
```


L.7.5 Schleifen – Schleifenvermeidung – Laufzeitvergleich

```
##################################################
# Loesungsvorschlag zu Blatt 7 Aufgabe 5
##################################################

##################################################
# Teil (a)
##################################################
for.schleife <- function(a,b){
    n <- length(a)
    m <- length(b)
    s <- numeric(m)

    for(i in 1:m)
        for(j in 1:n){
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
            if(a[j] <= b[i]) s[i] <- s[i]+1
        }
        #zweite for-Schleife sehr viel kürzer
        #und schneller:
        #s[i] <- sum((a<=b[i]))

    return(s)
}

while.schleife <- function(a,b){
    n <- length(a)
    m <- length(b)
    s <- numeric(m)

    iteri <- 0
    while(iteri < m){
        iteri <- iteri + 1
```


```
        iterj ← 0
        while(iterj < n){
            iterj ← iterj + 1
            if (a[iterj] <= b[iteri])
                s[iteri] ← s[iteri]+1
        }
    }

    return(s)
}

repeat.schleife ← function(a,b){
    n ← length(a)
    m ← length(b)
    s ← numeric(m)

    iteri ← 0
```


```
    repeat{
        iteri ← iteri + 1
        iterj ← 0
        repeat{
            iterj ← iterj + 1
            if (a[iterj] <= b[iteri])
                s[iteri] ← s[iteri]+1
            if (iterj == n) break
        }
        if(iteri == m) break
    }

    return(s)
}

keine.schleife ← function(a,b){
    at ← table(a)
```


```
    M <- outer(as.numeric(names(at)), b, "<=")
    s <- as.vector(apply(M*as.vector(at), 2, sum))

    return(s)
}


##################################################
# Teil (b)
##################################################
vergleich <- function(a, b, verfahren="keine"){
    switch(verfahren,
        for.schleife=for.schleife(a,b),
        while.schleife=while.schleife(a,b),
        repeat.schleife=repeat.schleife(a,b),
        keine.schleife=keine.schleife(a,b))
}
```


```
##################################################
# Teil (c)
##################################################
a ← rpois(5000, lambda=8)
b ← rpois(50, lambda=12)

system.time(erg.for ← vergleich(a=a,b=b,
                    verfahren="for.schleife"))
system.time(erg.while ← vergleich(a=a,b=b,
                    verfahren="while.schleife"))
system.time(erg.repeat ← vergleich(a=a,b=b,
                    verfahren="repeat.schleife"))
system.time(erg.keine ← vergleich(a=a,b=b,
                    verfahren="keine.schleife"))
```


L.7.6 Adaptives Verfahren zur 2-dimensionalen numerischen Integration

```
##################################################
# Loesungsvorschlag zu Blatt 7 Aufgabe 6
##################################################

##################################################
# Teil (a)
##################################################
area2 <- function(f, a, b, c, d, M=10, limit=10,
                  iter=1, eps=1e-5, ...)
{
#     print(iter)
    x1 <- runif(M, min=a, max=b)
    y1 <- runif(M, min=c, max=d)
    int1 <- (b-a)*(d-c)*mean(f(x1,y1,...))
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
#     cat("int1:\t", int1, "\n")

    x2 <- runif(M, min=a, max=b)
    y2 <- runif(M, min=c, max=d)
    int2 <- (b-a)*(d-c)*mean(f(x2,y2,...))
#     cat("int2:\t", int2, "\n")

    if(abs(int1-int2) < eps) return((int1+int2)/2)

    if(limit==0)
    {
        warning("Maximale Rekursionstiefe erreicht!")
        return((int1+int2)/2)
    }

    if((iter%%2)==0)
    {
```


```
        m <- (c+d)/2
        (Recall(f, a, b, c, m, limit=limit-1,
                iter=iter+1, eps=eps, ...)+
        Recall(f, a, b, m, d, limit=limit-1,
                iter=iter+1, eps=eps, ...))
    }
    else
    {
        m <- (a+b)/2
        (Recall(f, a, m, c, d, limit=limit-1,
                iter=iter+1, eps=eps, ...)+
        Recall(f, m, b, c, d, limit=limit-1,
                iter=iter+1, eps=eps, ...))
    }

}
```


```
#########################################################
# Teil (b)
#########################################################

fbeta.tmp <- function(x,y, alph, bet)
{
    assign("valx", c(valx, x), envir=sys.frame(0))
    assign("valy", c(valy, y), envir=sys.frame(0))
    if(x^2+y^2 > 1)
    {
        return(0)
    }
    else
    {   ww<-sqrt(x^2+y^2)^(alph-1)
        ww<-ww*(1-sqrt(x^2+y^2))^(bet-1)
        return(ww)
    }
```


```
}

assign("valx", NULL, envir=sys.frame(0))
assign("valy", NULL, envir=sys.frame(0))
system.time(int <- area2(fbeta.tmp, a=-1, b=1, c=-1,
                         d=1, M=10, limit=10, alph=3,
                         bet=2))
par(pch="*")
plot(valx, valy, type="p",
     xlab="x-Achse", ylab="y-Achse")
title("Auswertungsstellen der Integration")
```


L.8 Lösungsvorschläge Blatt 8

L.8.1 Visualisierung

```
##########################################################
# Loesungsvorschlag zu Blatt 8 Aufgabe 1
##########################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

##########################################################
```


```
# Teil (a)
##########################################################
# Blatt 6 Aufgabe 3
n <- 1000
U <- matrix(runif(n=2*n, min=0, max=1),
            nrow=n, ncol=2)


##########################################################
# Teil (b)
##########################################################
K <- ((U[,1]-0.5)^2 + (U[,2]-0.5)^2 <= 0.25)
pi4 <- sum(K)/n
pi.emp <- 4*pi4


##########################################################
```


```
# Teil (c)
##################################################
# Normalapproximation:
# sqrt(n) (p_n-p) -> N(0,p(1-p))
# mit p_n relativer Anteilswert;
# konservativ: p(1-p)<=1/4

# P( |p_n-p|> 10^-6)=
#      P( sqrt(n)|p_n-p|> sqrt(n) 10^-6)
#      ~ P(|N(0,1/4)|> sqrt(n) 10^-6)  !<= 0.95
# <=> 2 Phi(2 sqrt(n) 10^-6)-1 >= 0.95
# <=> Phi(2 sqrt(n) 10^-6) >= 0.975
# <=> 2 sqrt(n) 10^-6 >= Phi^-1(0.975)
# <=> sqrt(n) >= 2Phi^-1(0.975) 10^6
# <=> n >= 4 x 10^12

##################################################
```


```
# Berechnung und Plot des Kreises
##################################################
x <- seq(from=0, to=1, by=0.001)
y1 <- 0.5 + sqrt(x-x^2)
y2 <- 0.5 - sqrt(x-x^2)
li <- -0.2
re <- 1.3
par(col=1)
plot(x, y1, type="l", xlim=c(li,re), ylim=c(li,re),
     xlab="x-Achse", ylab="y-Achse", lwd=2)
lines(x, y2, lwd=2)

# Plot des Quadrates
lines(c(0,1,1,0,0), c(0,0,1,1,0), lwd=2)

# Plot der Punkte
im.Kreis <- U*K
```


```
im.Kreis ← matrix(im.Kreis[im.Kreis!=0],
                  nrow=sum(K), ncol=2)
aus.Kreis ← U*(1-K)
aus.Kreis ← matrix(aus.Kreis[aus.Kreis!=0],
                  nrow=sum(1-K), ncol=2)
par(col=2, pch=1)
points(im.Kreis[,1], im.Kreis[,2])
par(col=3, pch=4)
points(aus.Kreis[,1], aus.Kreis[,2])

par(col=1)
title(paste("Schätzung der Kreiszahl pi =",
        round(pi, 6)))
legend(li, re, legend=c("im Kreis", "ausserhalb"),
        pch=c(1,4), col=c(2,3))
text(0.5, -0.1,
    labels=paste("Der aktuelle Schätzwert für pi ist"
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
    pi.emp))
#dev.off()


##########################################################
# Teil (b)
##########################################################
# Blatt 7 Aufgabe 2
luecke.vec <- function(n) { #erst die x Koordinate
    X.x <- runif(n)

# erzeuge eine Matrix mit identischen Zeilen
    X.xx1 <- matrix(X.x, nrow=n, ncol=n)
# Matrtix mit identischen Spalten
    X.xx2 <- t(X.xx1)

# Matrix mit Eintraegen (X_i-X_j)^2_{i,j}
```


```
    X.xd ← (X.xx1-X.xx2)^2

# Diagonale aus Minimumbildung ausschliessen
    X.xd[row(X.xd)==col(X.xd)] ← 2

# Platz machen
    rm(X.xx1,X.xx2)

# das gleiche fuer die y Koordinate
    X.y ← runif(n)
    X.yy1 ← matrix(X.y, nrow=n, ncol=n)
    X.yy2 ← t(X.yy1)
    X.yd ← (X.yy1-X.yy2)^2
    X.yd[row(X.yd)==col(X.yd)] ← 2

# Platz machen
    rm(X.yy1,X.yy2)
```


```
# Addition der Koordinaten
    X.d ← sqrt(X.xd + X.yd)

# Minimum und Minimalstelle
    mi ← min(X.d)
    ind ← X.d==mi
    ind[row(ind)>col(ind)] ← FALSE
    ind.x ← cumsum(apply(ind, 1, sum))
    ind.x1 ← sum(ind.x==0) + 1
    ind.y ← cumsum(apply(ind, 2, sum))
    ind.y1 ← sum(ind.y==0) + 1

# modifiziert:
#   png("BL8Aufg1.png")
#   win.metafile("BL8Aufg1.wmf")
```


```
    plot(X.x,X.y)
    lines(c(X.x[ind.x1], X.x[ind.y1]),
          c(X.y[ind.x1], X.y[ind.y1]), col="green")
    return(list(mat1=X.x, mat2=X.y,
           ind1=ind.x1,ind2=ind.y1))
#   dev.off()

    return(min(sqrt(X.d)))
}

erg<-luecke.vec(20)

##################################################
# Teil (c)
##################################################
n <- 50
M <- 10
```


```
# leider gibt es unseres Wissens nach
# keine for/while-freie Moeglichkeit
# festzustellen, ob es in einem logischen
# Vektor X Folgen der Laenge m von "TRUE"
# gibt daher:
gibtruns <- function(X, m){
    i <- 0
    w1 <- 0
    wm <- FALSE
    n <- length(X)
    while((i<n)&&(wm==FALSE)){
        i<-i+1
        w1<-ifelse(X[i]==TRUE, w1+1, 0)
        wm <- (w1>=m)
    }
```


```
    return(wm)
}

walk.1 <- function(n){
    schritt <- sample(c(-1, 1), n, replace=T)
    pfad <- cumsum(schritt)
}

mwalks.1 <- function(X, n){mwalk.1(n)}
Pfade.1 <- function(M, n){sapply(1:M, mwalk.1, n=n)}
erg <- Pfade(M, n)

rwalk <- function(n, M, folg=4){
    Pfade.x <- Pfade(M, n)
    Pfade.y <- Pfade(M, n)

    missM <- numeric(M)
```


```
    missM[1:M] ← NA

    xli ← min(Pfade.x) − 1
    xre ← max(Pfade.x) + 1
    yli ← min(Pfade.y) − 1
    yre ← max(Pfade.y) + 1

# Test im ersten Quadranten
    pf.im1 ← (Pfade.x>0)*(Pfade.y>0)
    vier.spalte.im1 ← apply(pf.im1, 2, gibtruns,
                            m=folg)>0
# ergibt Indikator welche der Spalten/Pfade 4
# mal aufeinanderfolgend im 1. Quadranten

# Test im zweiten Quadranten
    pf.im2 ← (Pfade.x<0)*(Pfade.y>0)
    vier.spalte.im2 ← apply(pf.im2, 2, gibtruns,
```


```
                                    m=folg)>0

# Test im dritten Quadranten
    pf.im3 <- (Pfade.x>0)*(Pfade.y<0)
    vier.spalte.im3 <- apply(pf.im3, 2, gibtruns,
                                    m=folg)>0

# Test im vierten Quadranten
    pf.im4 <- (Pfade.x<0)*(Pfade.y<0)
    vier.spalte.im4 <- apply(pf.im4, 2, gibtruns,
                                    m=folg)>0


    vier.spalte <- pmax(vier.spalte.im1,
       vier.spalte.im2, vier.spalte.im3,
       vier.spalte.im4)

# jittern und Spalten auswaehlen sowie
```


```
# diese durch Missings trennen
  Pfade4.xm <- rbind(jitter(Pfade.x[,vier.spalte==1],
                    amount=0.2), missM[vier.spalte==1])
  Pfade4.ym <- rbind(jitter(Pfade.y[,vier.spalte==1],
                    amount=0.2), missM[vier.spalte==1])
#   print(Pfade4.xm)
    pfzahl <- (dim(Pfade4.xm))[2]

#   png(paste(PFAD,"BL6Aufg1_2.png",
#       collapse="", sep=""))
#   win.metafile(filename=paste(PFAD,"BL6Aufg1_2.wmf",
#       collapse="",sep=""))

    plot(c(xli,xre), c(yli,yre), type="n",
         xlab="", ylab="")
# entweder alle auf einmal
```


```
#    lines(Pfade4.xm, Pfade4.ym)

# oder als Schleife mit verschiedenen Farben
    i ← 0
    while(i<pfzahl){
        i←i+1
        lines(Pfade4.xm[,i], Pfade4.ym[,i], col=i)
    }
#    dev.off()
    return(pfzahl)
}

rwalk(n=40, M=10, folg=4)
rwalk(n=100, M=100, folg=30)
```


L.8.2 Bundestagswahl 2005

```
##################################################
# Loesungsvorschlag zu Blatt 8 Aufgabe 2
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

Filename <- "wahlen2.txt"
wahl <- read.table(file = Filename, row.names = 1,
                   header = TRUE, skip = 1, sep = "\t")
```


```
print(wahl)
wahl <- rbind(wahl, wahl[3,]-apply(wahl[4:8,],
        2, sum))
row.names(wahl)[9] <- "Sonstige"
wahl <- rbind(wahl, wahl[1,]-wahl[2,])
row.names(wahl)[10] <- "Nichtwaehler"
print(wahl)

##########################################################
# Teil (b)&(c)
##########################################################
# png("BL8Aufg2.png")
# win.metafile("BL8Aufg2.wmf")

def.par <- par(no.readonly = TRUE)
# zu layout: mat-Parameter regelt, in welcher
# Reihenfolge die Panels bemalt werden; wenn
```


```
# eine Zahl 2x aufaucht heisst das, dass die
# entsprechenden Panels vereinigt werden und
# beplottet werden
layout(mat=matrix(c(1,2,1,3), ncol=2))

matplot(log(t(wahl[4:10, ])), type="l",
        col=c("red", "black", "green", "yellow",
              "pink", "grey", "blue"), lwd=2,
        xlab="Bundesland", ylab="Waehlerzahl",
        axes=F, lty=1, ylim=c(log(1e3),
        log(2.5*10e6)))
axis(1, at=1:6, labels=names(wahl)[1:6])
axis(2, at=log(c(1e3, 1e4, 1e5, 1e6, 5e6)),
     labels=as.character(c(1e3, 1e4, 1e5, 1e6, 5e6)))
box()
title(paste("Bundestagswahl 2005: ",
      "ausgewaehlte Bundeslaender"))
```


```
legend(1, log(1e4), legend=row.names(wahl)[4:10],
       ncol=3, cex=0.6,
       fill=c( "red", "black", "green",
               "yellow", "pink", "grey", "blue"))

# absolute Zahlen in farbigen Bar-Plots
names(wahl) <- c("BAY", "B-W", "B", "SAC",
                 "NRW","GES")
wahl.gruppen <- as.matrix(wahl[4:10, ])
# Unterscheidung mit Farben
loc <- barplot(wahl.gruppen,
                 col=c("red","black","green","yellow",
                       "purple","grey","blue"),
                        beside=T)
total <- apply(wahl.gruppen, 2, sum)
# text(loc, total + par("cxy")[2], total, cex=0.5,
# xpd=T)
```


```
title("Ausgewaehlte Bundeslaender", cex=2)
# print(locator(1))
if (interactive()==T){
    legend(locator(1),legend=row.names(wahl.gruppen),
           ncol=2, cex=0.5, fill=c("red", "black",
           "green", "yellow", "purple", "grey",
           "blue"))
   }
else{
    legend(xy,3.5, max(wahl.gruppen),
        legend=row.names(wahl.gruppen), ncol=2,
           cex=0.5, fill=c("red", "black", "green",
               "yellow", "purple", "grey", "blue"))}

# relative Zahlen in farbigen Bar-Plots
# Unterscheidung mit Schraffuren
# alle bis auf Nichtwaehler beziehen sich
```


```
# auf gueltige Stimmen
wahl.gruppen.rel <- t(t(wahl.gruppen)/
        as.vector(t(as.matrix(wahl[3,]))))*100
# Nichtwaehler beziehen auf alle Berechtigten
wahl.gruppen.rel[7,] <- t(t(wahl.gruppen[7,])/
        as.vector(t(as.matrix(wahl[1,]))))*100
print(wahl.gruppen.rel)
loc <- barplot(wahl.gruppen.rel, angle=(1:7)*360/7,
                density=(1:7)*5, col=c("red", "black",
                "green", "yellow", "purple", "grey",
                "blue"))
total <- apply(wahl.gruppen.rel, 2, sum)
# text(loc, total + par("cxy")[2], total, cex=0.5,
# xpd=T)
title("Ausgewaehlte Bundeslaender", cex=2)
legend(1.5, 30, legend=row.names(wahl.gruppen),
        ncol=2, cex=0.5,
```


```
        fill=c("red", "black", "green",
        "yellow", "purple", "grey", "blue"))

#dev.off()
```


Peter Ruckdeschel
Matthias Kohl

L.8.3 Multivariate Konvexkombination

```
##################################################
# Loesungsvorschlag zu Blatt 8 Aufgabe 3
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

n ← 100

##################################################
# Teil (a)
```


```
##################################################
k <- 2 #Dimension
r <- 0.1 #Kontaminationsradius
muid <- c(0,0)
mucont <- c(1,1)
Sid <- matrix(c(1,-1,-1,2), nrow = k, ncol = k)
Scont <- diag(rep(0.64,k))
Y <- matrix(rnorm(k*n), nrow = k, ncol = n)
Xid <- Sid %*% Y + muid
Xcont <- Scont %*% Y + mucont
# "Protokoll": Ub == 1 <=> Kontamination
Ub <- rbinom(n = n, size = 1, prob = r)
Ub <- rbind(Ub,Ub)
Xb <- (1-Ub)*Xid + Ub*Xcont
Xbcont <- Ub*Xcont
Xbcont <- matrix(Xbcont[Xbcont != 0], nrow = 2)
```


```
##################################################
# Teil (b)
##################################################
r <- qnorm(1-0.025)
zerl <- svd(Sid%*%Sid)
theta <- seq(from = 0, to = 2*pi, by = 0.01)
X <- rbind(r*cos(theta)*sqrt(zerl$d[1]),
           r*sin(theta)*sqrt(zerl$d[2]))
X <- zerl$u %*% X

# Grenzen für die Plots
re <- ceiling(max(Xb[1,], X[1,]))
li <- floor(min(Xb[1,], X[1,]))
ob <- ceiling(max(Xb[2,], X[2,]))
un <- floor(min(Xb[2,], X[2,]))
```


```
# png("BL8Aufg3.png")
# win.metafile("BL8Aufg3.wmf")

par(mfrow = c(1,1), col = "dark red")
plot(X[1,], X[2,], type = "l", xlab = "X1",
     ylab = "X2", xlim = c(li,re),
     ylim = c(un,ob), lwd = 2)



##########################################################
#Teil (c)
##########################################################
Xbu <- zerl$u %*% Xb
ind1 <- ((Xbu[1,]^2/zerl$d[1] +
        Xbu[2,]^2/zerl$d[2]) <= r^2)
#innerhalb der Ellipse
Xid.i1 <- ind1*Xb[1,]
```


```
Xid.i2 ← ind1*Xb[2,]
Xid.i1 ← Xid.i1[Xid.i1 != 0]
Xid.i2 ← Xid.i2[Xid.i2 != 0]
#ausserhalb der Ellipse
Xid.a1 ← (1-ind1)*Xb[1,]
Xid.a2 ← (1-ind1)*Xb[2,]
Xid.a1 ← Xid.a1[Xid.a1 != 0]
Xid.a2 ← Xid.a2[Xid.a2 != 0]

Xbcontu ← zerl$u %*% Xbcont
ind2 ← ((Xbcontu[1,]^2/zerl$d[1] +
        Xbcontu[2,]^2/zerl$d[2]) <= r^2)
#innerhalb der Ellipse
Xbcont.i1 ← ind2*Xbcont[1,]
Xbcont.i2 ← ind2*Xbcont[2,]
Xbcont.i1 ← Xbcont.i1[Xbcont.i1 != 0]
Xbcont.i2 ← Xbcont.i2[Xbcont.i2 != 0]
```


```
#ausserhalb der Ellipse
Xbcont.a1 ← (1-ind2)*Xbcont[1,]
Xbcont.a2 ← (1-ind2)*Xbcont[2,]
Xbcont.a1 ← Xbcont.a1[Xbcont.a1 != 0]
Xbcont.a2 ← Xbcont.a2[Xbcont.a2 != 0]

par(col = "black")
plot(0, 0, type = "n", xlab = "X1", ylab = "X2",
     xlim = c(li,re), ylim = c(un,ob))
par(col = "green")
points(Xid.i1, Xid.i2)
par(col = "blue")
points(Xid.a1, Xid.a2)
par(col = "red")
points(Xbcont.i1, Xbcont.i2)
par(col = "orange")
points(Xbcont.a1, Xbcont.a2)
```


```
par(col = "black")
title(expression(paste("Bivariate Konvexkombination ",
        (1-r), italic(N), "(", mu[id], ",", S[id]^2, ") ",
        + r, italic(N), "(", mu[cont], ",", S[cont]^2,
        ")")))
text(0.5, 3, "95% Konfidenzintervall", col = "dark red",
        cex = 1.1)
legend(li, un+4, legend = c("ideal innerhalb",
        "ideal ausserhalb", "kont innerhalb",
        "kont ausserhalb"), pch = 1,
        col = c("green", "blue", "red", "orange"))
legend(li, un+4, legend = c("ideal innerhalb",
        "ideal ausserhalb", "kont innerhalb",
        "kont ausserhalb"), pch = 1,
        col = c("green", "blue", "red", "orange"))
#dev.off()
```


L.8.4 Regressionsplots

```
##################################################
# Loesungsvorschlag zu Blatt 8 Aufgabe 4
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

##################################################
# Teil (a)
##################################################
```


```
n ← 16
eps ← rnorm(16, sd = 2)
X ← -5:10
theta ← 2
alpha ← 1
Y ← theta*X+alpha+eps

##################################################
# Teil (b)
##################################################

Z ← exp(Y)
plot(X, Z, type = "p", ylim = c(min(Z),
     max(Z,exp(theta*X+alpha))),
     xlab = "X", ylab = "Z")

# png("BL8Aufg4_1.png")
```


```
par(new = TRUE)
plot(X, exp(theta*X+alpha), type = "l",
     ylim = c(min(Z), max(Z,exp(theta*X+alpha))),
     xlab = "X", ylab = "Z")
#dev.off()

##################################################
# Teil (c)
##################################################

# png("BL8Aufg4_2.png")

windows()
plot(X, Y, type = "p", xlim = c(-5, 10),
     axes = FALSE, ylim = c(min(Y), max(Y)),
     xlab = "X", ylab = "")
```


```
axis(2, at = c(-10, -5, 0, 5, 10, 15, 20, 25),
     labels = c("-10", "-5", "0", "5", "10",
                "15", "20", "25"))
axis(4, at = c(-10, -5, 0, 5, 10, 15, 20, 25),
     labels = c("4e-5", "7e-3", "1", "1.5e2",
                "2.2e4", "3.27e7", "4.85e8",
                "7.20e10"))
axis(1, at = c(-5, -1, 0, 1, 5, 10),
      labels = c(-5, -1, 0, 1, 5, 10))
box()

par(new = T)
plot(X, theta*X+alpha, type = "l",
      xlim = c(-5, 10), axes = FALSE,
      ylim = c(min(Y), max(Y)),
      xlab = "X", ylab = "")
title("Nicht-lineare Regression")
```


```
#dev.off()
```


L.9 Lösungsvorschläge Blatt 9

L.9.1 3D-Plot

```
#########################################################
# Loesungsvorschlag zu Blatt 9 Aufgabe 1
#########################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

####
```


```
### Funktion die den Black-Scholes Preis eines
### europaeischen Calls zur Zeit t berechnet

BScall <- function(t, T, Xt, K, r, sigma)
### K Ausuebungspreis
### T Ausuebungszeitpunkt
### r risikoloser Zins (logarithmisch)
### sigma Volatilitaet
### Xt Aktienkurs zum Zeitpunkt t
{dt <- T-t
 K1 <- K/Xt
 h  <- (-r * dt + log(K1))/sqrt(dt)/sigma
 p0 <- pnorm(-h + sigma * sqrt(dt) / 2)
 p1 <- pnorm(-h - sigma * sqrt(dt) / 2)
 return( Xt * p0 - K * p1 * exp(-r * dt) )
}
```


```
## sigma , t Gitter

sigmas ← seq(0.05, 0.6, length = 100)
ts ← seq(0, 0.8, length = 100)
### nicht verlangt:
Ks ← seq(8, 15, length = 100)

BSgitter.st ← outer(ts, sigmas, BScall, T = 0.8,
                    r = 0.05, K = 10, Xt = 12)
BSgitter.sK ← outer(Ks, sigmas, BScall, T = 0.8,
                    r = 0.05, t = 0.79, Xt = 12)


?image
image(ts, sigmas, BSgitter.st)
image(ts, sigmas, BSgitter.st,
      main = "Black-Scholes Callpreis",
      xlab = expression(sigma), ylab = "t",
```

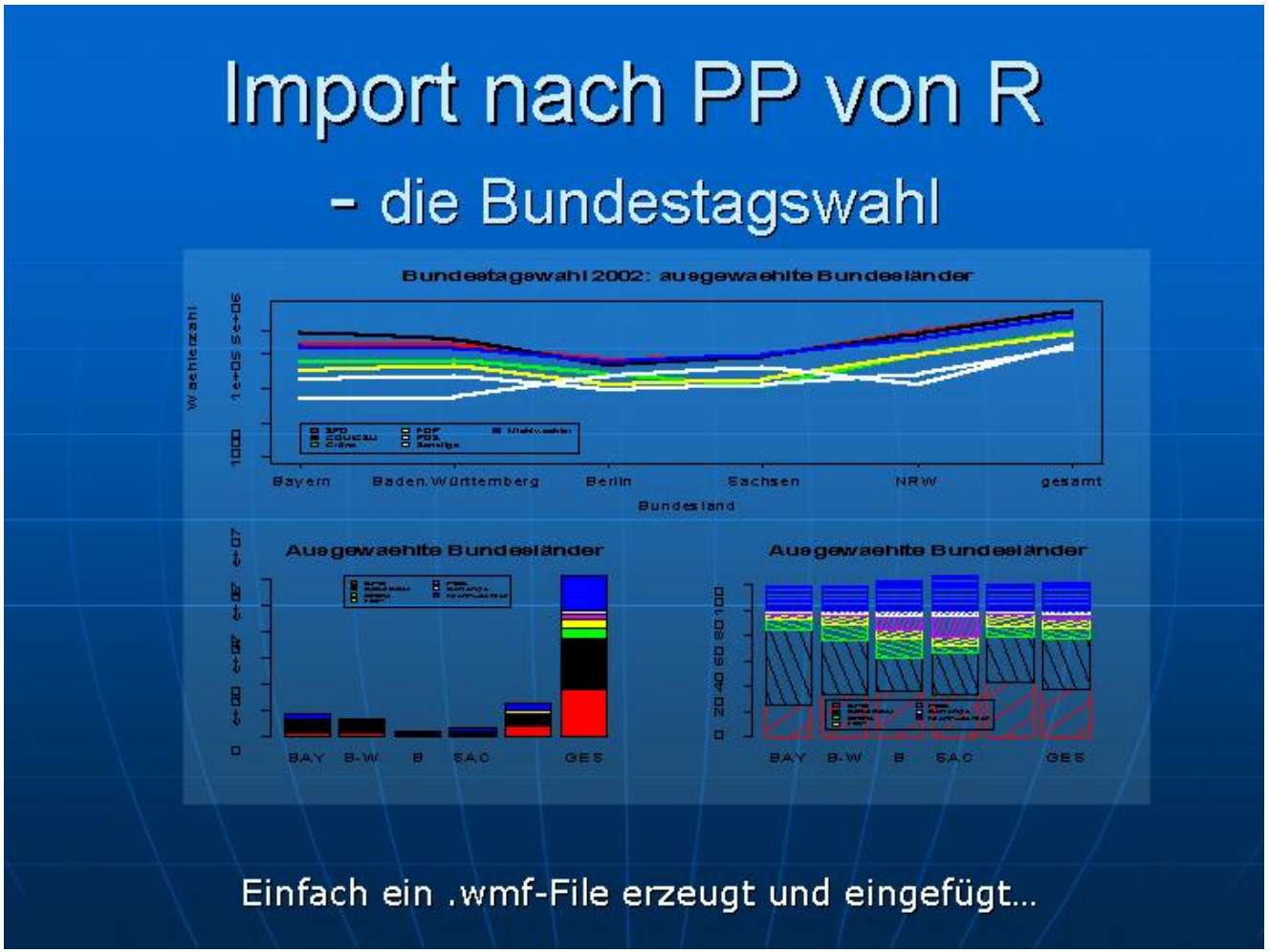

```
      col = heat.colors(200))

contour(ts, sigmas, BSgitter.st,
        levels = seq(0, 6, length = 35),
        add = TRUE, col = "black")


?persp
persp(ts, sigmas, BSgitter.st)

persp(ts, sigmas, BSgitter.st, theta = -40,
      zlab = "Black-Scholes Callpreis",
      ylab = expression(sigma), xlab = "t",
      shade = 0.5, ticktype = "detailed",
      col = "lightgreen")

persp(Ks, sigmas, BSgitter.sK, theta = 40,
```


```
        zlab = "Black-Scholes Callpreis",
        ylab = expression(sigma),  xlab = "K",
        shade = 0.5, ticktype = "detailed",
        col = "lightgreen")

?layout

## Ziel:
# 1 1 1 1
# 2 3 4 4
win.metafile(file = "Testfile.wmf")

layout(matrix( c(rep(1,4),2,3,4,4),
                2, 4, byrow = TRUE))

### jetzt die vier Graphiken
image(ts, sigmas, BSgitter.st,
```


```
        main = "Black-Scholes Callpreis",
        xlab = expression(sigma), ylab = "t",
        col = heat.colors(200))

contour(ts,sigmas,BSgitter.st,
          levels = seq(0, 6, length = 35),
          add = TRUE, col = "black")
#
persp(ts, sigmas, BSgitter.st, theta = -40,
        zlab = "Black-Scholes Callpreis",
        ylab = expression(sigma), xlab = "t",
        shade = 0.5, ticktype = "detailed",
        col = "lightgreen")
#
persp(Ks, sigmas, BSgitter.sK, theta = 40,
        zlab = "Black-Scholes Callpreis",
        ylab = expression(sigma),  xlab = "K",
```


```
        shade = 0.5, ticktype  =  "detailed",
        col = "lightgreen")
#
image(Ks,sigmas,BSgitter.st,
        main = "Black-Scholes Callpreis",
        xlab = expression(sigma), ylab = "K",
        col = heat.colors(200))

contour(Ks,sigmas,BSgitter.st,
          levels = seq(0, 8, length = 35),
          add = TRUE, col = "black")

##################################################
## layout zuruecksetzen
layout(matrix(1,1,1))
### File schliessen
dev.off()
```


L.9.2 Powerpoint-Präsentation



L.9.3 Farben

```
##################################################
# Loesungsvorschlag zu Blatt 9 Aufgabe 3
##################################################


require(RColorBrewer)
    ### im Zweifelsfall noch installieren ...

## zB
mypalette <- brewer.pal(7,"PiYG")
display.brewer.pal(7,"PiYG")

require(RColorBrewer)  ### im Zweifelsfall noch installieren ...

## zB
```


Peter Ruckdeschel
Matthias Kohl

```
mypalette←brewer.pal(7,"PiYG")
display.brewer.pal(7,"PiYG")
## zur Verfügung stellen:

?palette
palette(mypalette)
plot(1:7,rep(1,7),pch=paste(1:7),col=1:7,cex=3)
palette(rev(mypalette))
plot(1:7,rep(1,7),pch=paste(1:7),col=1:7,cex=3)
```


L.9.5 Chernoff-Gesichter

```
##################################################
# Loesungsvorschlag zu Blatt 9 Aufgabe 5
##################################################


require(alr3)
require(aplpack)

data(banknote)
?banknote #
## liefert:

## 6 Messungen an 100 echten und 100 gefälschten
## Schweizer Banknoten
## der Datensatz ist von Dimension 200 x 7,
```


```
## wobei die Merkmalsdimensionen sind:

#  +"Length":    Länge der Note in mm
#  +"Left":      Breite der linken Seite in mm
#  +"Right":     Breite der rechten Seite in mm
#  +"Bottom":    unterer Rand in mm
#  +"Top":       oberer Rand in mm
#  +"Diagonal":  Länge der Bild-Diagonalen in mm
#  +"Y":         Indikator 0=echt 1=gefälscht

#ur-Quelle:
#     Flury, B. and Riedwyl, H. (1988):
#     /Multivariate Statistics: A practical approach./
#     London: Chapman & Hall.


##erste 10 echte und falsche:
```


```
faces(banknote[c(1:10,101:110),])

###zufällig

randindex ← sample(1:200,30)
labindex ← rep("falsch",30)
labindex[ randindex<100 ] ← "echt"
faces(banknote[randindex,], label = labindex)
```


Peter Ruckdeschel
Matthias Kohl

L.9.6 R und TclTk

```
##################################################
# Loesungsvorschlag zu Blatt 9 Aufgabe 6
##################################################

require(tcltk)
demo(tkdensity)
file.show(system.file(package = "tcltk",
                      "demo/tkdensity.R"))

### dieses File haben wir leicht modifiziert

local({ ### um globale Variablen anschliessend
        ### wieder gut loeschen zu koennen

### Aus Blatt 9 Aufgabe 1
```


```
BScall ← function(t, T, Xt, K, r, sigma)
{dt ← T-t
 K1 ← K/Xt
 h  ← (-r * dt + log(K1))/sqrt(dt)/sigma
 p0 ← pnorm(-h + sigma * sqrt(dt) / 2)
 p1 ← pnorm(-h - sigma * sqrt(dt) / 2)
 return( Xt * p0 - K * p1 * exp(-r * dt) )
}

###

## neu plot:
plotCall ← function(T = 1, K = 0.4, Xt = 0.5,
                    r = 0.05, sigma = 0.3)
    {t0 ← seq(0, T, length = 100)
     plot(t0, BScall(t0, T = T, K = K, Xt = Xt,
```


```
                    r = r, sigma = sigma),
          type = "l", lwd = 1.5,   xlab = "",
          ylab = "")
    title(main =
"Black-Scholes Preis für einen Europäischen Call")
    title(sub = substitute(
       paste(T==T0, "~~", "  ", K==K0, "  ", "  ",
          ~~X[t]==Xt0, "  ", "  ", ~~r==r0, "%,  ",
          ~~sigma==sigma0, "%"),
       list(T0 = T, K0 = K, Xt0 = Xt, r0 = r*100,
          sigma0 = sigma*100)),
     line = 3)
    title(xlab = "Zeit t",  ylab = "Preis des Calls" ,
     line = 2)}

plotCall3D <- function(T = 1, K = 0.4, Xt = 0.5,
                    r = 0.05, theta = -40)
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
{ ts ← seq(0, T, length = 50)
  sigmas ← seq(0, 0.5, length = 50)
  BSgitter.st ↞ outer(ts, sigmas, BScall, T = T,
                      r = r, K = K, Xt = Xt)
  persp(ts, sigmas, BSgitter.st, theta = theta,
        zlab = "Black-Scholes Callpreis",
        ylab = expression(sigma), xlab = "t",
        shade = 0.5, ticktype = "detailed",
        col = "lightgreen")
  title(sub = substitute(
          paste(T==T0, ",  ", K==K0, ",  ",
                X[t]==Xt0, ",  ", r==r0, "%"),
          list(T0 = T, K0 = K, Xt0 = Xt,
                r0 = r*100)),
        line = 3)
}
```


Peter Ruckdeschel
Matthias Kohl

```
T.sav  <-  1
Xt.sav  <-  0.5
K.sav  <-  1
r.sav  <-  0.05
sigma.sav  <-  0.3
theta.sav  <-  -40
e2D3D.sav  <-  2

dum <-tclVar(3)
T.m <- tclVar(1)
Xt.m <- tclVar(0.5)
K.m <- tclVar(1)
K.m <- tclVar(1)
r.m <- tclVar(0.05)
sigma.m <- tclVar(0.3)
theta.m <- tclVar(-40)
e2D3D.m <- tclVar(2)
```


```
replot <- function(...) {
    T.sav <<- T <- as.numeric(tclObj(T.m))
    Xt.sav <<- Xt <- as.numeric(tclObj(Xt.m))
    K.sav <<- K <- as.numeric(tclObj(K.m))
    r.sav <<- r <- as.numeric(tclObj(r.m))
    sigma.sav <<- sigma <-
                as.numeric(tclObj(sigma.m))
    theta.sav <<- theta <-
                 as.numeric(tclObj(theta.m))
    e2D3D.sav <<- e2D3D <-
                 as.numeric(tclObj(e2D3D.m))

    if(e2D3D == 2)
        eval(substitute(
        plotCall(T = T, K = K, Xt = Xt, r = r,
                 sigma = sigma)
```


```
                    ))
    else
        eval(substitute(
        plotCall3D(T = T, K = K, Xt = Xt, r = r,
                   theta = theta)
                    ))
    }

replot.maybe <- function(...)
{
    if ((any(T.sav != as.numeric(tclObj(T.m))) ||
             Xt.sav != as.numeric(tclObj(Xt.m)) ||
             K.sav != as.numeric(tclObj(K.m)) ||
             r.sav != as.numeric(tclObj(r.m)) ||
             e2D3D.sav !=
               as.numeric(tclObj(e2D3D.m)))||
             ( sigma.sav !=
```


```
                    as.numeric(tclObj(sigma.m)) &&
                    e2D3D.sav == 2 ) ||
                  ( theta.sav !=
                    as.numeric(tclObj(theta.m)) &&
                    e2D3D.sav == 3
                    ))
      replot()
 }

tx ← c("Spitze", "toll", "prima")
spitze ← function(...){
  if (e2D3D.sav == 2)
    {my ← range(BScall(t = seq(0, T.sav, length = 200),
                       T = T.sav, K = K.sav,
                       Xt = Xt.sav, r = r.sav,
                       sigma = sigma.sav))
     mx ← c(0,T.sav)
```


```
  }else{
      my ← c(-0.5,0.5)
      mx ← c(-.5,.5)
  }
 dumv ← as.numeric(tclObj(dum))
  eval.parent(substitute(
      text(x = runif(1, min = mx[1], max = mx[2]),
           y = runif(1, min = my[1], max = my[2]),
           labels = tx[dumv], col = "red", cex = 1.5)
           ))
  }

base ← tktoplevel()
tkwm.title(base, "Black-Scholes")

spec.frm ← tkframe(base, borderwidth = 2)
left.frm ← tkframe(spec.frm)
```


```
right.frm ← tkframe(spec.frm)

frame1 ← tkframe(left.frm,  relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame1,
          text = "Ausübungszeitpunkt T"))
tkpack(tkscale(frame1,  command = replot.maybe,
               from = 0.05,  to = 3.00,
               showvalue = TRUE,
               variable = T.m,
               resolution = 0.02,
               orient = "horiz"))

frame2 ← tkframe(left.frm,  relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame2,
          text = "Aktienkurs Xt zum Zeitpunkt t"))
```


```
tkpack(tkscale(frame2,  command = replot.maybe,
               from = 0.05,  to = 10.00,
               showvalue = TRUE,  variable = Xt.m,
               resolution = 0.02,
               orient = "horiz"))

frame3 <-tkframe(left.frm,  relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame3,
                text = paste("Blatt 9 Aufgabe 6",
                 "Sie sind der Meinung das ist ",
                              sep="\n")))
for ( i in c(1, 2, 3) ) {
    tmp <- tkradiobutton(frame3, command = spitze,
                         text = tx[i], value = i,
                         variable = dum)
    tkpack(tmp,  anchor = "w")
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
}

frame4 <-tkframe(left.frm,  relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame4,
                text = "2D oder 3D-Plot"))

tx2D3D <- c("","2D","3D")
for ( i in c(2, 3) ) {
    tmp <- tkradiobutton(frame4, command = spitze,
                         text = tx2D3D[i],
                         value = i,
                         variable = e2D3D.m)
    tkpack(tmp,  anchor = "w")
}

frame5 <-tkframe(right.frm,  relief = "groove",
```


```
                    borderwidth = 2)
tkpack(tklabel (frame5,
          text = "Ausübungspreis (Strike) K"))
tkpack(tkscale(frame5,  command = replot.maybe,
               from = 0.05,  to = 10,
               showvalue = TRUE,  variable = K.m,
               resolution = 0.02,
               orient = "horiz"))

frame6 <-tkframe(right.frm,  relief = "groove",
                    borderwidth = 2)
tkpack(tklabel (frame6,  text = "risikoloser Zins r"))
tkpack(tkscale(frame6,  command = replot.maybe,
               from = 0.001,  to = 0.2,
               showvalue = TRUE,  variable = r.m,
               resolution = 0.001,  orient = "horiz"))
```


```
frame7 <-tkframe(right.frm, relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame7, text = "Volatilität sigma"))
tkpack(tkscale(frame7, command = replot.maybe,
               from = 0.001, to = 0.5,
               showvalue = TRUE,
               variable = sigma.m,
               resolution = 0.001,
               orient = "horiz"))

frame8 <-tkframe(right.frm, relief = "groove",
                 borderwidth = 2)
tkpack(tklabel (frame8, text = "Perspektivwinkel theta"))
tkpack(tkscale(frame8, command = replot.maybe,
               from = -180, to = 180,
               showvalue = TRUE,
               variable = theta.m,
```


```
                    resolution = 1,
                    orient = "horiz"))


    tkpack(frame1, frame2, frame3, frame4, fill = "x")
    tkpack(frame5, frame6, frame7, frame8, fill = "x")
    tkpack(left.frm,  right.frm, side = "left",
           anchor = "n")

    q.but ← tkbutton(base, text = "Quit",
                     command =
                       function()tkdestroy(base))


    tkpack(spec.frm,  q.but)
})
```


L.10 Lösungsvorschläge Blatt 10

L.10.1 Shapiro-Wilk, Kolmogorov-Smirnov, χ^2-Anpassungstest

```
##########################################################
# Loesungsvorschlag zu Blatt 10 Aufgabe 1
##########################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####
```


```
FILENAME ← "normal.txt"
#
normal ← read.table(file = FILENAME, header = TRUE)
attach(normal)
names(normal)



############################
###  Shapiro-Wilk-Test
############################
shapiro.test(x)
# => keineswegs ablehnen

#################################
###  Kolmogoroff-Smirnoff-Test
#################################
# Frage: NV mit welchen Parametern in K-S-Test?
```


```
# mit vorgegebenen (theoretischen)
ks.test(x, "pnorm",mean = 1,sd = sqrt(3))
# => ablehnen (hoch signifikant!)

# mit denen, so dass das Ablehnen am schwierigsten ist
ks2dist ← function(theta, XX){
    ks.test(XX, "pnorm", mean = theta[1],
            sd = theta[2])$statistic
}
kmin2dist ← function(XX, theta0 = c(0,1)){
    optim(par = theta0, fn = ks2dist, XX = XX)$par
}
th ← kmin2dist(x)
print(th)
ks.test(x, "pnorm", mean = th[1], sd = th[2])
# => keineswegs ablehnen
```


```
################
###  chi^2-Test
################
#Unterteilen der Daten in Klassen
#mit Hilfe von qnorm(p, 1, 3)
#p=0.1, 0.2, 0.3, ..., 0.9

p <- seq(from = 0, to = 1.0, by = 0.1)
x.class <- cut(x,
      qnorm(p, mean = 1, sd = sqrt(3)))
h.class <- table(x.class)
chisq.test(h.class)

# => nicht ablehnen

#oder wieder  mit den Parametern, so dass das
```


```
#Ablehnen am schwierigsten ist
chi2dist ← function(theta, XX){
    x.class ← cut(x, qnorm(p,
            mean = theta[1], sd = theta[2]))
    h.class ← table(x.class)
    chisq.test(h.class)$statistic
}
chmin2dist ← function(XX, theta0 = c(mean(XX),sd(XX))){
    optim(par = theta0, fn = chi2dist,
          XX = XX)$par
}

th ← chmin2dist(x)
print(th)
x.class ← cut(x,
     qnorm(p, mean = th[1], sd = th[2]))
h.class ← table(x.class)
```


```
chisq.test(h.class)
# => nicht ablehnen

##########
### qq-Plot
##########
qqnorm(x)
qqline(x)

detach()
```


Peter Ruckdeschel
Matthias Kohl

L.10.2 Wilcoxon und t-Test, χ^2- und F-Test

```
##################################################
# Loesungsvorschlag zu Blatt 10 Aufgabe 2
##################################################

##################################################
# Teil (a)
##################################################
## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME<-"uscomp.txt"
```


```
uscomp ← read.table(FILENAME, header = TRUE)
attach(uscomp)
names(uscomp)
X6.liste ← split(X6, Sector)

wilcox.test(X6.liste$Energy, X6.liste$Finance,
            conf.level = 0.9)
#  => nicht ablehnen !

t.test(X6.liste$Energy, X6.liste$Finance,
       conf.level = 0.9)
# => nicht ablehnen !

##################################################
# Teil(b)
##################################################
```


```
# chi^2-Test
alpha <- 0.1
sigma0 <- 100
n <- length(X6.liste$Energy)
ccritu <- qchisq(p = (alpha/2), df = n-1)
ccrito <- qchisq(p = (1-alpha/2), df = n-1)

Teststatistik <- (n-1)*var(X6.liste$Energy)/sigma0
pwert <- 2*pchisq(q = Teststatistik, df = n-1)

cat("chi^2-Test zum Signifikanzniveau von alpha=",
    alpha, "%:\n")
cat("unterer krit. Wert:\t", ccritu, "\n")
cat("oberer krit. Wert:\t", ccrito, "\n")
cat("Pruefgroesse:\t", Teststatistik, "\n")
cat("P-Value:\t",pwert,"\n")
# => nicht ablehnen !
```


```
##################################################
# Teil (c)
##################################################
# F-Test
var.test(X6.liste$Energy, X6.liste$Finance,
         conf.level = 0.9)
# => nicht ablehnen !

detach()
```


L.10.3 Fisher- und t-Test

```
##################################################
# Loesungsvorschlag zu Blatt 10 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################
## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME<-"kredit1.txt"
```


```
kredit ← read.table(file = FILENAME, header = TRUE)
attach(kredit)
names(kredit)

# Berufstatus == 1: arbeitslos
Berufstatus ← as.integer(Berufstatus == 1)
# Zusammenfassen der "Kreditwuerdigen"
# nach dem Berufstatus
job ← as.matrix(table(Kreditwuerdigkeit,
        Berufstatus))

# Anteil der "Kreditwuerdigen" bzw.
# "Kreditunwuerdigen"
ja ← 1-job[1,1]/sum(job[1,])
nein ← 1-job[2,1]/sum(job[2,])
cat("Anteil der Kreditwuerdigen:\t",
```


```
    ja, "\n")
cat("Anteil der Kreditunwuerdigen:\t",
    nein, "\n")


##################################################
# Teil (b)
##################################################
alpha <- 0.1 #Signifikanzniveau

kredit[,5] <- Berufstatus
# Aufspalten des Kreditdatensatzes nach
# Kreditwuerdigkeit
kredit.liste <- split(kredit,
                      Kreditwuerdigkeit)
kredit0 <- kredit.liste[[1]][,5]
kredit1 <- kredit.liste[[2]][,5]
```


```
#t-Test
t.test(kredit0, kredit1,
       conf.level = (1-alpha))

#asymptotischer Fisher-Test
#Vertauschen der Spalten in job-Matrix,
#um die Anteile der Arbeitslosen zu testen
job1 <- job[,2]
job1 <- cbind(job1, job[,1])
prop.test(job1, conf.level = (1-alpha))

#exakter Fisher-Test
ccritu <- qhyper(p = (alpha/2), m = sum(job[1,]),
           n = sum(job[2,]), k = sum(job[,2]))

ccrito <- qhyper(p = (1-alpha/2), m = sum(job[1,]),
```


```
                n = sum(job[2,]), k = sum(job[,2]))

cat("unterer krit. Wert:\t", ccritu, "\n")
cat("oberer krit. Wert:\t", ccrito, "\n")
cat("Pruefgroesse:\t", job[1,2], "\n")

detach()
```


L.10.4 Testvergleich durch Simulation

```
##################################################
# Loesungsvorschlag zu Blatt 10 Aufgabe 4
##################################################

n ← 10
alpha ← 0.05

##################################################
# freiwilliger Teil:
##################################################
# NPT
# Dichtquotient :
# Dichte der X1..Xn unter H0
#  prod_i (dnorm(xi))
#        = (2pi)^(-n/2) exp(-sum_i(x_i^2)/2)
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
#
# Dichte der X1..Xn unter H1
#  prod_i (dnorm(xi-1))
#        = (2pi)^(-n/2) exp(-sum_i((x_i-1)^2)/2)
#
# log DQ: 1/2 sum_i x_i^2-(x_i-1)^2 = sum_i x_i - n/2
                        = n (mean(X)-1/2)
# also DQ >= c <=> mean(X) >= c'
#
# aber mean(X)~ [unter H0] N(0,1/n)
#
# => critischer Wert als P(N(0,1/n)>c')=alpha,
# => P(N(0,1)<=c'sqrt(n))=1-alpha
# => c'=qnorm((1-alpha))/sqrt(n)
c <- qnorm(1-alpha)/sqrt(n)
# c = 0.5201
```


```
##########################################################
# VZT
#
# betrachte Yi = 1(Xi>0);
# dann Yi uiv ~ [unter H0] Bin (1,p0)
# und p0 = 1/2
# schreibe
# P_0 fuer P unter H0 und P_1 fuer P unter H1
# E_0          "          E_1          "
# => N ~ [unter H0] Bin(n,p0), p0 = 1/2
# => P_0(N = k) = dbinom(x = k, size = n,
#                        prob = 1/2)
#
p0 <- 1/2
#
# unter H1 wird im allgemeinen N groesser sein!
# => Test: lehne ab falls N zu gross;
```


```
# Fehler 1. Art !<= alpha
# <=> 1-pbinom(q = c, size = n, prob = 1/2) <= alpha
# <=> pbinom(q = c, size = n, prob = 1/2) >= 1-alpha
# Niveau wird nicht ausgeschoepft:
# Ablehnen bei N = 8,9,10
#       ^= pbinom(q = 7, size = 10, prob = 1/2) = 0.9453125
# Ablehnen bei N = 9,10
#       ^= pbinom(q = 8, size = 10, prob = 1/2) = 0.9892578
# => unrandomisiert Ablehnen bei 9,10
# => randomisiert: immer Ablehnen bei 9,10,
#          bei 8 mit Ws r0 (wie auf Angabe) ablehnen
qbinom(1-alpha, n, p0)
# 8 !
pbinom(q = 7, size = n, prob = p0)
pbinom(q = 8, size = n, prob = p0)
r0 <- (pbinom(8, n, p0)-0.95)/dbinom(8, n, p0)
#
```


Peter Ruckdeschel
Matthias Kohl

```
# MACHT
#
# Macht VZT unrandomisiert:
#
# E_1[1(N = 9 oder 10)]  = P_1(N = 9 oder 10)
# N ~[unter H1] Bin(n,p1), p1 = P(N(1,1)>0)
#
##################################################
# Bemerkung: nun Testsituation zweier
# Muenzen M0,M1
# P(M0 = 1) = 1/2, P(M1 = 1) = 0.841
# nach n Wuerfen mit Muenze, die entweder
# M0 oder M1 ist
# Entscheidung: War's M0 oder M1
##################################################
p1 <- 1 - pnorm(0, mean = 1)          # 0.8413
#
```


```
macht.vztu ← 1 − pbinom(8, n, p1)
### 0.5129
#
# Macht VZT randomisiert:
#
# E_1[1(N = 9 oder 10)+1(N = 8)1(R = 1)] =
# = P_1(N = 9 oder 10)+P_1(N = 8)P(R = 1) =
# = p1+r0 dbinom(8,n,p1)
macht.vztr ← macht.vztu + r0 ∗ dbinom(8, n, p1)
### 0.7669
#
# Macht NPT
#
macht.npt ← 1 − pnorm(c, mean = 1, sd = 1/sqrt(n))
### 0.9354
#
##################################################
```


Peter Ruckdeschel
Matthias Kohl

```
xa ← c(0.1, 2.1, −1.2, 1.0, 1.6, 1.3, 0.2, 0.1,
       0.3, 2.2)
meana ← mean(xa) # 0.77  => NPT lehnt ab
pvalue.npt ← 1 − pnorm(meana, sd = 1/sqrt(n))
### 0.0074
Na ← sum(xa>0)   # 9     => VZTu VZTr lehnen ab
pvalue.vzt ← 1 − pbinom(q = 8, size = n, prob = p0)
### 0.0107

xac ← c(−20.2, 2.1, −1.2, 1.0, 1.6, 1.3, 0.2, 0.1,
        0.3, 2.2)
meanac ← mean(xac) # −1.26  => NPT lehnt nicht ab
pvaluec.npt ← 1 − pnorm(meanac,sd = 1/sqrt(n))
### 1.0000
Nac ← sum(xac>0)   # 8     => VZTu lehnt ab
pvaluec.vztu ← 1 − pbinom(q = 7, size = n, prob = p0)
```


```
### 0.0547
R ← rbinom(1, 1, r0)
R # falls R = 1 ablehnen, falls R = 0 annehmen...
#
#Begruendung:
#
# in NPT gehen die Beobachtungen unbeschraenkt ein
#  => beliebige Auslenkung der Teststatistik
#     durch eine Beobachtung
# in VZT gehen nur die Vorzeichen ein
#  => keine beliebige Auslenkung der Teststatistik
#     durch eine Beobachtung
##################################################


# eigentliche Aufgabe
##################################################
```


```
# Teil (a)
##########################################################

M <- 10000
ve <- 0.05

X.id.0 <- matrix(rnorm(M*n), nrow = M,
                 ncol = n)
X.id.1 <- matrix(rnorm(M*n, mean = 1),
                 nrow = M, ncol = n)



##########################################################
# Teil (b)
##########################################################
#NPT
entsch.id.0.npt <- apply(X.id.0, 1, mean) > c
```


```
empfehler.id.1.npt ← mean(entsch.id.0.npt)
cat("Empirischer Fehler 1. Art bei NPT
     (ideale Situation):", empfehler.id.1.npt, "\n")
entsch.id.1.npt ← apply(X.id.1,1,mean) > c
empfehler.id.2.npt ← 1 − mean(entsch.id.1.npt)
cat("Empirischer Fehler 2. Art bei NPT
     (ideale Situation):", empfehler.id.2.npt, "\n")

#VZTu (unrandomisiert)
entsch.id.0.vztu ← apply((X.id.0>0),1,sum) > 8
empfehler.id.1.vztu ← mean(entsch.id.0.vztu)
cat("Empirischer Fehler 1. Art bei VZTu
     (ideale Situation):", empfehler.id.1.vztu, "\n")
entsch.id.1.vztu ← apply((X.id.1>0),1,sum) > 8
empfehler.id.2.vztu ← 1 − mean(entsch.id.1.vztu)
cat("Empirischer Fehler 2. Art bei VZTu
     (ideale Situation):", empfehler.id.2.vztu, "\n")
```


```
#VZTr (randomisiert)
R.0 <- matrix(rbinom(M,1,r0), nrow = M, ncol = 1)
# Aussondern der Ereignisse N = 8
entsch.id.0.vztr0 <- apply((X.id.0>0),1,sum) == 8
entsch.id.0.vztr <- entsch.id.0.vztu
                        + entsch.id.0.vztr0 * R.0
empfehler.id.1.vztr <- mean(entsch.id.0.vztr)
cat("Empirischer Fehler 1. Art bei VZTr
     (ideale Situation):", empfehler.id.1.vztr, "\n")

R.1 <- matrix(rbinom(M,1,r0), nrow = M, ncol = 1)
# Aussondern der Ereignisse N = 8
entsch.id.1.vztr0 <- apply((X.id.1>0),1,sum) == 8
entsch.id.1.vztr <- entsch.id.1.vztu
                        + entsch.id.1.vztr0 * R.1
empfehler.id.2.vztr <- 1 - mean(entsch.id.1.vztr)
```


```
cat("Empirischer Fehler 2. Art bei VZTu
     (ideale Situation):", empfehler.id.2.vztr,"\n")


##################################################
# Teil (c)
##################################################
# nenne Y Xc;
# es gilt Yc0~(1-ve)N(0,1)+ve N(10,1)
#         Yc1~(1-ve)N(1,1)+ve N(-10,1)
U.0 <- matrix(rbinom(M*n, 1, ve), nrow = M, ncol = n)
U.1 <- matrix(rbinom(M*n, 1, ve), nrow = M, ncol = n)
X.c.0 <- X.id.0+U.0*10
X.c.1 <- X.id.1-U.1*10


##################################################
```


Peter Ruckdeschel
Matthias Kohl

```
# Teil(d)
##########################################################
#NPT
entsch.c.0.npt ← apply(X.c.0, 1, mean)>c
empfehler.c.1.npt ← mean(entsch.c.0.npt)
cat("Empirischer Fehler 1. Art bei NPT
        (kont. Situation):", empfehler.c.1.npt, "\n")
entsch.c.1.npt ← apply(X.c.1, 1, mean)>c
empfehler.c.2.npt ← 1 − mean(entsch.c.1.npt)
cat("Empirischer Fehler 2. Art bei NPT
        (kont. Situation):", empfehler.c.2.npt, "\n")

#VZTu (unrandomisiert)
entsch.c.0.vztu ← apply((X.c.0>0), 1, sum)>8
empfehler.c.1.vztu ← mean(entsch.c.0.vztu)
cat("Empirischer Fehler 1. Art bei VZTu
        (kont. Situation):", empfehler.c.1.vztu, "\n")
```


```
entsch.c.1.vztu ← apply((X.c.1>0), 1, sum)>8
empfehler.c.2.vztu ← 1 − mean(entsch.c.1.vztu)
cat("Empirischer␣Fehler␣2.Art␣bei␣VZTu
␣␣␣␣␣␣␣␣(kont.␣Situation):", empfehler.c.2.vztu, "\n")

#VZTr (randomisiert)
R.0 ← matrix(rbinom(M, 1, r0), nrow = M, ncol = 1)
# Aussondern der Ereignisse N = 8
entsch.c.0.vztr0 ← apply((X.c.0>0), 1, sum) == 8
entsch.c.0.vztr ← entsch.c.0.vztu
                        + entsch.c.0.vztr0 * R.0
empfehler.c.1.vztr ← mean(entsch.c.0.vztr)
cat("Empirischer␣Fehler␣1.Art␣bei␣VZTr
␣␣␣␣␣␣␣␣(kont.␣Situation):", empfehler.c.1.vztr, "\n")

R.1 ← matrix(rbinom(M, 1, r0), nrow = M, ncol = 1)
# Aussondern der Ereignisse N = 8
```


```
entsch.c.1.vztr0 ← apply((X.c.1>0), 1, sum) == 8
entsch.c.1.vztr ← entsch.c.1.vztu
                        + entsch.c.1.vztr0 * R.1
empfehler.c.2.vztr ← 1 - mean(entsch.c.1.vztr)
cat("Empirischer Fehler 2. Art bei VZTu
         (kont. Situation):", empfehler.c.2.vztr, "\n")
```


L.11 Lösungsvorschläge Blatt 11

L.11.1 Indiskrete Umfrage

```
##################################################
# Loesungsvorschlag zu Blatt 11 Aufgabe 1
##################################################

##################################################
# M1i: Ergebnis des 1. Muenzwurfs bei i-tem Befragtem
# M2i: Ergebnis des 2. Muenzwurfs bei i-tem Befragtem
# Ii:  Antwort auf indiskr. Frage bei i-tem Befragtem
# Ai:  gegebene Antwort
#
# "Kopf" ^= 1,  "Zahl" ^= 0
#
```


```
#  M1i, M2i, Ii sto unabh
#  M1i ~Bin(1,1/2)
#  M2i ~Bin(1,1/2)
#  Ii  ~Bin(1,p),  p gesucht
#
# (M1i, M2i, Ii)_i identisch verteilt
#
# => Ai=M1i*Ii+(1-M1i)*M2i  uiv
#
#    Ai nimmt ebenfalls nur die Werte 0,1 an
#
#   also Ai ~Bin(1,q)
#   dabei
#   q=q(p)=
#    =P(Ai=1)=
#    =P((M1i=1 und Ii=1) oder (M1i=0 und M2i=1))=
#        dabei oder ^= entweder oder
```


Peter Ruckdeschel
Matthias Kohl

```
#    also =P(M1i=1 und Ii=1)+P(M1i=0 und M2i=1)=
#           sto. Unabh. von M1i, Ii und M1i, M2i
#         =P(M1i=1)P(Ii=1)+P(M1i=0)P(M2i=1)=
#         =p/2+1/4
##################################################

##################################################
# ML--Schaetzung
##################################################
# Likelihood von A_1...A_n
#
#   P_p( Ai=ai, i=1..n ) =[sto u]  prod_i P_p(Ai=ai)
# = prod_i [q(p)^(1{Ai=1})+(1-q(p))^(1{Ai=0})]
#           S= #{i: ai=1}
# = q(p)^S+(1-q(p))^(n-S)
# also
#   log P_p( A_i=a_i, i=1..n )
```


```
# = S log (q(p)) - (n-S) log (1-q(p))
#
# und damit
#
#   Lambda_p((Ai)_i)
# = d/dp log P_p( A_i=a_i , i=1..n )
# = S/q * 1/2 - (n-S)/(1-q) * 1/2
#
# und damit d/dp log P_p( A_i=a_i , i=1..n ) = 0
# <=> q^ML=S/n <=> p^ML = 2 S/n - 1/2
#
# dabei S ~ Bin(n,q)
#
# E_p[p^ML]= 2 E_p[S]/n-1/2 = 2q-1/2 = p
# also p^ML biasfrei
# Var_p[p^ML]= 4/n^2 Var_p[S]= 4q(1-q)/n
#
```


```
# Fisher-Info:
#   E_p[(Lambda_p((Ai)_i))^2]=
# = E_p[{(S-nq)/(2q(1-q))}^2]=
# = n/(4(q(1-q)) = 1/Var_p[p^ML]
#
# also erreicht p^ML die Cramer-Rao-Schranke
#
##########################################################

##########################################################
# Normalapproximation fuer p^ML
##########################################################
# de Moivre-Laplace
#
#     sqrt(n) (S/n-q) => N(0,q(1-q))
#
#=>   sqrt(n) 2(S/n-q) => N(0,4q(1-q))
```


```
#
#aber 2(S/n-q) = p^ML-p
##########################################################

##########################################################
# approximatives Konfidenzintervall
##########################################################
#
# P_p(|p^ML-p| < s) ~= P(|N(0,1)|
#          < s/2 sqrt(n/(q(1-q)) ) != 0.9756
# sei t=s/2 sqrt(n/(q(1-q)))
# P(-t <= N(0,1) <= t) = Phi(t)-Phi(-t)
#          = 2Phi(t)-1 != 0.9756
# also Phi(t)!= 1.9756/2
# <=>  t=Phi^(-1)(1.9756/2)
# <=>  s=2 Phi^(-1)(1.9756/2) sqrt(q(1-q)/n)
# und Konf.intervall : [p^ML -s, p^ML +s]
```


```
#
# Vergleich: "unverschleierte" Befragung
#
# s0= Phi^(-1)(1.9756/2) sqrt(p(1-p)/n)
#
##################################################################



##################################################################
# eigentliche Berechnungen
##################################################################
n ← 2300
S ← 682
pML ← 2*S/n - 1/2
qML ← S/n
s ← 2*qnorm(1.9756/2)*sqrt(qML*(1-qML)/n)
s0 ← qnorm(1.9756/2)*sqrt(pML*(1-pML)/n)
```


```
fac ← s/s0
cat("Schaetzung␣fuer␣p:",pML)
cat("\n97.56%-Konfidenzintervall:␣[",pML-s,";",
    pML+s,"]")
cat("\n97.56%-Konfidenzintervall␣(bei
␣␣␣␣unverschleierter␣Befragung):␣[",
    pML-s0,";",pML+s0,"]")
cat("\n^=um␣Faktor␣",fac,"␣␣groesser␣und␣um␣Faktor␣",
        fac^2, "mehr␣Beobachtungen␣noetig␣fuer␣gleiche
␣␣␣␣␣␣␣␣Genauigkeit")
```


L.11.2 ML–Schätzer für K aus $\mathrm{HypGeo}(N, K, n)$

```
##################################################
# Loesungsvorschlag zu Blatt 11 Aufgabe 2
##################################################

##################################################
# Umrechnung der Parameter in Angabe (Suffix a)
# in Parameter in R (Suffix R)
#
# Na = mR +nR
# Ka = mR
# na = kR
# ka = xR
#
##################################################
```


Peter Ruckdeschel
Matthias Kohl

```
Na <- 30
na <- 10
ka <- c(3,4,6,4,7)

#moegliche Werte fuer K
Km <- 0:30

##################################################
# Likelihood als Funktion in Km
# da wir es in sapply einsetzen wollen:
# als Argument X in Likel
##################################################
Likel <- function(X, N, n, k){
    kR <- n
    xR <- k
    mR <- X
    nR <- N - X
```


```
    p ← dhyper(x=xR, m=mR, n=nR, k=kR)

    return(prod(p))
}

# Berechnung des Likelihoodvektors
LV ← sapply(Km, Likel, N=Na, n=na, k=ka)
i ← which.max(LV)
KML ← Km[i]
print(KML)

##################################################
# Erweiterung: Bestimmung der Genauigkeit
# mit Bootstrap
##################################################
resampl ← function(X, k, N, n, Km){
    le ← length(k)
```


```
    ks <- sample(k, le, replace=T)
    LV <- sapply(Km, Likel, N=Na, n=na, k=ks)

    return(Km[which.max(LV)])
}

zahlbootstrap <- 1000
kboot <- sapply(1:zahlbootstrap, resampl,
              k=ka, N=Na, n=na, Km=Km)
Kmb <- mean(kboot)
Kmv <- var(kboot)
cat("Mittelwert:", Kmb, "\tVarianz:", Kmv, "\n\n")
```


L.11.3 Simulationsstudie

```
##################################################
# Loesungsvorschlag zu Blatt 11 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################

# implementiere summe_i psi_theta(x_i)
# mit psi_m,theta(x)= max(-m,min((x-theta),m))
# (vergleiche Uebung)
psim <- function(theta, y, m0=0.7){
            sum(pmin(pmax(-m0,y-theta),m0))/length(y)
# schneller als mean()!
}
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
# finde Nullstelle theta_0 so dass
# summe_i psi_{m,theta_0}(x_i) = 0
Mschaetz <- function(x, m=0.7){
    uniroot(psim, low=-20, up=20, tol=1e-10,
            y=x, m0=m, maxiter=20)$root
}

##################################################
# Teil (b) -- (e)
##################################################
# will wieder mit sapply arbeiten => X statt n

tuealles <- function(X, M, ve=0.1, m0=0.7){
#zur besseren Lesbarkeit
    n <- X
```


```
# Teil (b)
    Xid <- matrix(rnorm(M*n), nrow=M, ncol=n)

# Auswahlmechanismus wie in Blatt 2
    R <- matrix(rbinom(n=M*n, size=1, p=ve),
                nrow=M, ncol=n)
    Xct <- matrix(rcauchy(M*n), nrow=M, ncol=n)
    Xc <- (1-R)*Xid + R*Xct

# Teil (c)
# Schaetzerauswertung
    Xqid <- apply(Xid, 1, mean)
    Medid <- apply(Xid, 1, median)
    Mid <- apply(Xid, 1, Mschaetz, m=m0)
    Sid <- cbind(Xq=Xqid, Med=Medid, M=Mid)

    boxplot(data.frame(Sid))
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
    title(main="ideale Situation",
          sub=paste("n=",n,collapse=""))

    Xqc <- apply(Xc, 1, mean)
    Medc <- apply(Xc, 1, median)
    Mc <- apply(Xc, 1, Mschaetz, m=m0)
    Sc <- cbind(Xq=Xqc, Med=Medc, M=Mc)

    boxplot(data.frame(Sc))
    title(main="kontam. Situation",
          sub=paste("n=",n,collapse=""))
    boxplot(data.frame(Sc[,2:3]))
    title(main="kontam. Situation",
          sub=paste("n=",n,collapse=""))

# Teil (d)
# ideale Situation
```


```
    asvarnXid ← n*var(Xqid)
    asBiasXid ← sqrt(n)*mean(Xqid)

    asvarnMedid ← n*var(Medid)
    asBiasMedid ← sqrt(n)*mean(Medid)

    asvarnMid ← n*var(Mid)
    asBiasMid ← sqrt(n)*mean(Mid)

    cat("Empirische Werte fuer Varianz, Bias
                und MSE: --- ideale Situation --- [ n =",
            n, "]\n\n")
    cat("n x Varianz:\tXq: ", asvarnXid, "\t\tMed: ",
            asvarnMedid, "\t\tM-Sch: ", asvarnMid, "\n")
    cat("n^0.5 x Bias:\t\tXq: ", asBiasXid, "\t\tMed: ",
            asBiasMedid, "\t\tM-Sch: ", asBiasMid, "\n")
    cat("n x MSE:\t\tXq: ", asvarnXid+asBiasXid^2, "\t\tMed: ",
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
                asvarnMedid+asBiasMedid^2, "\t\tM-Sch: ",
                asvarnMid+asBiasMid^2, "\n\n\n")

# kontaminierte Situation
    asvarnXc <- n*var(Xqc)
    asBiasXc <- sqrt(n)*mean(Xqc)

    asvarnMedc <- n*var(Medc)
    asBiasMedc <- sqrt(n)*mean(Medc)

    asvarnMc <- n*var(Mc)
    asBiasMc <- sqrt(n)*mean(Mc)

    cat("Empirische Werte fuer Varianz, Bias
 und MSE: --- kont. Situation --- [ n =",
                n, "]\n\n")
    cat("n x Varianz:\tXq: ", asvarnXc, "\t\tMed: ",
```


```
            asvarnMedc, "\t\tM-Sch: ", asvarnMc, "\n")
    cat("n^0.5 x Bias:\t\tXq: ", asBiasXc, "\t\tMed: ",
            asBiasMedc, "\t\tM-Sch: ", asBiasMc, "\n")
    cat("n x MSE:\t\tXq: ", asvarnXc+asBiasXc^2, "\t\tMed: ",
            asvarnMedc+asBiasMedc^2, "\t\tM-Sch: ",
            asvarnMc+asBiasMc^2, "\n\n\n")
}

# gesamter Ablauf
#FILENAME <- "BL11Aufg3plot.png"
M <- 1000
n <- c(5,10,100)
ve <- 0.1
m <- 0.7
# png(FILENAME, width=900, height=900)
par(new=F,mar=c(5.1,4.1,6.7,2.1))
par(mfrow=c(3,3))
```


```
g ← sapply(X=n, tuealles, M=M, ve=ve, m0=m)
par(mfrow=c(1,1))
par(new=F,mar=c(5.1,4.1,1.8,2.1))
mtext("Verschiedene␣Schaetzer␣in␣id.[N(0,1)]
␣␣␣␣␣␣␣␣␣&␣kont.Sit.[0.9N(0,1)+0.1Cauchy]")
#dev.off()
```


L.12 Lösungsvorschläge Blatt 12

L.12.1 Berechnung eines Quantils

```
##################################################
# Loesungsvorschlag zu Blatt 12 Aufgabe 1
##################################################

##################################################
# Teil (a)
##################################################
# Bisektionsalgorithmus
quantil.bisektion <- function(alpha=0.95, start=0,
                              ende=5, delta=1e-8){
    test <- -1.0
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
x ← start
x.li ← start
x.re ← ende
iter ← 0

while(abs(test) > delta){
    iter ← iter + 1
    if(test < 0){
        x.li ← x
        x ← (x.li + x.re)/2
    }
    else{
        x.re ← x
        x ← (x.li + x.re)/2
    }

    test ← pnorm(x)-alpha
```


```
    }

    cat("Iterationen fuer Bisektion:\t", iter, "\n")

    return(x)
}


##################################################
# Teil (b)
##################################################
# Newton-Algorithmus
quantil.newton <- function(alpha=0.95, start=1,
                           delta=1e-8){
    test <- 1.0
    x.alt <- start
```


```
    iter <- 0
    while(test > delta){
        iter <- iter + 1
        x.neu <- x.alt -
                 (pnorm(x.alt)-0.95)/dnorm(x.alt)
        test <- abs(x.neu-x.alt)
        x.alt <- x.neu
    }

    cat("Iterationen fuer Newton:\t", iter, "\n")

    return(x.alt)
}


##################################################
# Teil (c)
```


```
#########################################################
# Numerische Invertierung
x <- seq(from=0, to=5, by=0.01)
y <- pnorm(x)
quantil.invert <- splinefun(y,x)


#########################################################
# Iterationsalgorithmus
#########################################################
# Achtung: -pnorm(x.alt)!!!
quantil.iteration <- function(alpha=0.95,
                              start=1.5, delta=1e-8){
    test <- 1.0
    x.alt <- start

    iter <- 0
```


```
    while(test > delta){
        iter <- iter + 1
        x.neu <- -sign(dnorm(x.alt))*
            (pnorm(x.alt) - alpha) + x.alt
        test <- abs(x.neu-x.alt)
        x.alt <- x.neu
    }

    cat("Iterationen fuer Iteration:\t", iter, "\n")

    return(x.alt)
}


##########################################################
# Verwendung der Funktion uniroot
##########################################################
```


```r
fkt <- function(x, alpha){pnorm(x) - alpha}

# Zur Kontrolle
alpha <- 0.95
qnorm(alpha)
quantil.bisektion(alpha=alpha)
quantil.newton(alpha=alpha)
quantil.invert(alpha)
quantil.iteration(alpha=alpha)
uniroot(f=fkt, lower=0, upper=5, tol=1e-8,
        alpha=alpha)
```


L.12.2 Schätzung eines eindimensionalen Parameters

```
##################################################
# Loesungsvorschlag zu Blatt 12 Aufgabe 2
##################################################

##################################################
# Teil (a)&(b)
##################################################
# Hilfsfunktion zur Berechnung von Y_i!
# in der Maximum-Suche
factorial <- function(x){
    ifelse(x==0, erg <- 1,
           erg <- prod(seq(from=1, to=x, by=1)))

    return(erg)
```


Peter Ruckdeschel
Matthias Kohl

```
}

likelihood <- function(lambda,Y){
    1/(exp(lambda) - 1)^length(Y)*lambda^(sum(Y))/
    prod(sapply(Y, factorial))
}

nsgleichung<- function(lambda,Y){
    lambda-mean(Y)*(1-exp(-lambda))
}

#Maximum-Suche
maximum <- function(Y, start=1, ende=5,
                    delta=1e-4, kontrolle="keine"){
    n <- length(Y)

    #lambda-Gitter
```


```
lambda ← c(start, (ende-start)/2, ende)

gitter ← (ende-start)/2 #Gitterweite

test ← 1.0

maxalt ← -1
while((gitter > delta)||(test > delta)){
    lgit ←likelihood(lambda,Y)
    maxneu ← max(lgit)

    test ← abs(maxalt - maxneu)

    #Berechnung des neuen lambda-Gitters
    if(test > 0.0){
        maxalt ← maxneu
        mitte ← lambda[lgit==maxneu]
```


```
        gitter <- gitter/2
        lambda <- c(mitte-gitter, mitte,
                    mitte+gitter)
    }
    else{
        mitte <- lambda[lgit==maxneu]
        gitter <- gitter/2
        lambda <- c(mitte-gitter, mitte,
                    mitte+gitter)
    }

    switch(kontrolle, ausgabe=
        cat("Das aktuelle lambda:\t",
        mitte, "\n"), browser=browser())
}

return(mitte)
```


```
}

#Newton-Algorithmus
newton <- function(Y, start=2.5, delta=1e-4,
                   kontrolle="keine"){
    test <- 1.0
    lambda.alt <- start

    while(test > delta){
        lambda.neu <- lambda.alt - (lambda.alt
            - mean(Y)*(1-exp(-lambda.alt)))/
            (1-mean(Y)*exp(-lambda.alt))
        test <- abs(lambda.neu-lambda.alt)
        lambda.alt <- lambda.neu

        switch(kontrolle, ausgabe=
            cat("Das aktuelle lambda:\t",
```


```
            lambda.alt, "\n"), browser=browser())
    }

    return(lambda.alt)
}


#Bisektionsalgorithmus
bisektion <- function(Y, start=1, ende=5,
                      delta=1e-4, kontrolle="keine"){
    test <- 1.0
    test1 <- -1.0

    lambda <- start
    lambda.li <- start
    lambda.re <- ende
```


```
while(test > delta){
    if(test1 < 0){
        lambda.li ← lambda
        lambda ← (lambda.li + lambda.re)/2
    }
    else{
        lambda.re ← lambda
        lambda ← (lambda.li + lambda.re)/2
    }

    test1 ← lambda - mean(Y)*(1-exp(-lambda))
    test ← abs(test1)

    switch(kontrolle, ausgabe=
        cat("Das aktuelle lambda:\t",
        lambda, "\n"), browser=browser())
}
```


```
    return(lambda)
}


#####################################################
#Teil (c)
#####################################################
## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "truncpois.txt"
```


```
Y <- read.table(file = FILENAME, header = FALSE)
delta <- 1e-4

startw <- mean(Y[,1])
Z <- Y[,1]

lambda1 <- maximum(Y[,1], delta=delta,
                   kontrolle="ausgabe")
nlikelihood <- function(lambda, Y){
    -likelihood(lambda,Y)
}
optim(startw, nlikelihood, Y=Z)

lambda2 <-optim(startw, nlikelihood, Y=Z)$par

lambda3 <- bisektion(Y[,1], delta=delta,
                   kontrolle="ausgabe")
```


```
lambda4 ← newton(Y[,1], delta=delta,
                 kontrolle="ausgabe")
lwerte ← seq(0, max(Y[,1]), 0.03)
lbwerte ← sapply(lwerte, nsgleichung, Y=Z)
plot(lwerte, lbwerte)
interv ← c(1, max(Y[,1]))
lambda5 ← uniroot(nsgleichung, Y=Z,
          interval=interv,tol=delta)$root

 c(lambda1,lambda2,lambda3,lambda4,lambda5)
```


L.12.3 Numerische Probleme mit dem Coupon-Collector

```
##################################################
# Loesungsvorschlag zu Blatt 12 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################
Siebformel <- function(N, n){
    if(N==n){
        k <- 0:n
        erg <- (-1)^k*choose(n, k)*(1-k/n)^N
    }
    else{
        k <- 0:(n-1)
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
        erg <- (-1)^k*choose((n-1), k)
        erg <- erg*(1-(k+1)/n)^(N-1)
    }

    return(max(0, sum(erg)))
}

Siebformel.erw <- function(n){
    k <- 1:(n-1)
    erg <- (-1)^(k-1)*choose(n, k)*(1-k/n)^n/k

    return(n*(sum(erg)+1))
}

##################################################
# nicht Bestandteil der Aufgabe
##################################################
```


```
Siebformel.var <- function(n){
    k <- 1:(n-1)
    erg0 <- (-1)^(k-1)*choose(n, k)*(1-k/n)^n/k
    erg1 <- sum(erg0)
    erg2 <- (-1)^(k-1)*choose(n, k)*(1-k/n)^n/k^2

    erg <- (2*sum(erg2)-erg1^2)*n^2 - n*erg1

    return(erg)
}

n <- c(2, 3, 5, 20, 50, 75)
erw <- numeric(length(n))
streu <- numeric(length(n))

par(mfrow=c(3,2))
iter <- 0
```


```
for(i in n){
    WS <- numeric(10*i-i+1)
    for(j in seq(from=i, to=10*i, by=1))
        WS[j-i+1] <- Siebformel(N=j, n=i)

    iter <- iter + 1
    erw[iter] <- Siebformel.erw(i)
    streu[iter] <- sqrt(Siebformel.var(i))

    print(WS)
    plot(seq(from=i, to=length(WS)+i-1, by=1),
            WS, type="s", xlab="N", ylab="WS")
    title(paste("Wahrscheinlichkeitsfunktion
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣fuer␣n=", i))
}

cat("Die␣Erwartungswerte␣lauten:\t", erw, "\n")
```


```
cat("Die Streuung ist:\t", streu, "\n")
cat("Verhaeltnis E(N)/n:\t", erw/n, "\n")

# Logarithmisches Wachstum
# Zum Vergleich:
cat("Vergleich mit log(n)+0.6:\t", log(n)+0.6, "\n")
# Also fuer n=576 gilt naeherungsweise:
# erw/n in [6.5, 7]
# Also Erwartungswert: erw in [3744, 4032]

##########################################################
# Teil (b)
##########################################################
n ← 576
x ← 1:n
iter ← 100
counter ← rep(n, iter)
```


```
# zur Veranschaulichung:
# 30 Ziehungen aus 1:20
sample(1:20,30,replace=T)
# Tabelle der angenommen Werte
table(sample(1:20,30,replace=T))
# als Matrix
as.matrix(table(sample(1:20,30,replace=T)))
# Album voll <=> Matrix hat n Zeilen

for(i in 1:iter){
    test <- 1
    y <- sample(x, n, replace=T)
    while(test<n){
        counter[i] <- counter[i] + 1
        y <- c(y, sample(x,1,replace=T))
        y <- as.matrix(table(y))
```


```
        y ← as.integer(row.names(y))
        test ← length(y)
    }
    print(i)
}

cat("Mittelwert:\t", mean(counter), "\n")
cat("Empirische Varianz:\t", var(counter), "\n")

##################################################
# Teil (c)
##################################################
# E exakt
Eexakt ← function(n){n*sum(1/(1:n))}
Emacheron ← function(n){n*(log(n)+0.5772)}
n ← c(2, 3, 5, 20, 50, 75, 576)
sapply(n, Eexakt)
```


```
sapply(n, Emacheron)

##################################################
# Teil (d)
##################################################
ziehgeom1 <- function(n, N){
    rgeom(1,1-(n-1)/N)+1
} # Ziehung von X_n
ziehgeomn <- function(a, N){
    geoms<-sapply(1:N,ziehgeom1,N=N);sum(geoms)
}
ziehgeomo <- function(M, N){
    sapply(1:M,ziehgeomn,N=N)
}
```


Peter Ruckdeschel
Matthias Kohl

L.12.4 Optimale Prognose

```
##################################################
# Loesungsvorschlag zu Blatt 12 Aufgabe 4
##################################################

##################################################
# Teil (a)
##################################################

y <- seq(from=0, to=10, by=0.1)

bed.Erw <- function(y, Fkt, ...)
{
    integrand1 <- function(x, y, Fkt, ...)
    {
        eval(parse(text=paste("x*",Fkt,
```


```
                "(y-x, ...)*dnorm(x)", sep="")))
}
int1 <- integrate(integrand1, lower=-Inf,
            upper=Inf, rel.tol=1e-12, y=y,
            Fkt=Fkt, ...)$value

integrand2 <- function(x, y, Fkt, ...)
{
    eval(parse(text=paste(Fkt,
            "(y-x, ...)*dnorm(x)",
             sep="")))
}

int2 <- integrate(integrand2, lower=-Inf,
            upper=Inf, rel.tol=1e-12, y=y,
            Fkt=Fkt, ...)$value
```


```
    return(int1/int2)
}

f.konv <- function(x, r=0.1, sd=3)
{
    (1-r)*dnorm(x) + r*dnorm(x, sd=3)
}

g.i <- sapply(y, bed.Erw, Fkt="dnorm")
g.ii <- sapply(y, bed.Erw, Fkt="f.konv")
g.iii <- sapply(y, bed.Erw, Fkt="dcauchy")


##################################################
# Teil (b)
##################################################
```


```
postmod ← function(y, x.li=0, x.re=5,
                   weite=0.01, Fkt, ...)
{
    x ← seq(from=x.li, to=x.re, by=weite)
    eval(parse(text=paste("hilf ← ", Fkt,
                 "(y-x, ...)*dnorm(x)", sep="")))
    postmod ← x[order(hilf)][length(x)]

    return(postmod)
}

h.i ← sapply(y, postmod, Fkt="dnorm")
h.ii ← sapply(y, postmod, Fkt="f.konv")
h.iii ← sapply(y, postmod, Fkt="dcauchy")
```


```
##################################################
# Teil (c)
##################################################

g.i.glatt <- spline(y, g.i, n=5*length(y))
g.ii.glatt <- spline(y, g.ii, n=5*length(y))
g.iii.glatt <- spline(y, g.iii, n=5*length(y))
h.i.glatt <- spline(y, h.i, n=5*length(y))
h.ii.glatt <- spline(y, h.ii, n=5*length(y))
h.iii.glatt <- spline(y, h.iii, n=5*length(y))

par(col="green")
plot(g.i.glatt$x, g.i.glatt$y, type="l",
     xlim=c(min(y), max(y)),
     ylim=c(0,5), xlab="", ylab="")
par(new=T, col="blue")
plot(g.ii.glatt$x, g.ii.glatt$y, type="l",
```


```
        xlim=c(min(y), max(y)),
        ylim=c(0,5), xlab="", ylab="")
par(new=T, col="purple")
plot(g.iii.glatt$x, g.iii.glatt$y, type="l",
        xlim=c(min(y), max(y)),
        ylim=c(0,5), xlab="", ylab="")
par(new=T, col="yellow", lty=2)
plot(h.i.glatt$x, h.i.glatt$y, type="l",
        xlim=c(min(y), max(y)),
        ylim=c(0,5), xlab="", ylab="")
par(new=T, col="orange", lty=1)
plot(h.ii.glatt$x, h.ii.glatt$y, type="l",
        xlim=c(min(y), max(y)),
        ylim=c(0,5), xlab="", ylab="")
par(new=T, col="red")
plot(h.iii.glatt$x, h.iii.glatt$y, type="l",
        xlim=c(min(y), max(y)),
```


```
        ylim=c(0,5), xlab="", ylab="")
par(new=T, col="black")
plot(0, 0, type="n", xlim=c(min(y), max(y)),
     ylim=c(0,5), xlab="", ylab="")
title("Bedingter Erwartungswert und Posterior Modus")
legend(min(y), 5, legend=c("(a),i", "(a),ii",
       "(a),iii", "(b),i", "(b),ii", "(b),iii"),
       ncol=2, fill=c("green", "blue", "purple",
       "yellow", "orange", "red"))


##################################################
# Teil (d)
##################################################


n ← 5000
```


```
X ← rnorm(n)
eps.i ← rnorm(n)
r ← rbinom(n, prob=0.1, size=1)
eps.ii ← (1-r)*rnorm(n) + r*rnorm(n, sd=3)
eps.iii ← rcauchy(n)

Y.i ← X + eps.i
Y.ii ← X + eps.ii
Y.iii ← X + eps.iii

g.i.fkt ← splinefun(g.i.glatt$x, g.i.glatt$y)
g.ii.fkt ← splinefun(g.ii.glatt$x, g.ii.glatt$y)
g.iii.fkt ← splinefun(g.iii.glatt$x, g.iii.glatt$y)
h.i.fkt ← splinefun(h.i.glatt$x, h.i.glatt$y)
h.ii.fkt ← splinefun(h.ii.glatt$x, h.ii.glatt$y)
h.iii.fkt ← splinefun(h.iii.glatt$x, h.iii.glatt$y)
```


```
k.a.i.i ← g.i.fkt(Y.i)
k.a.i.ii ← g.i.fkt(Y.ii)
k.a.i.iii ← g.i.fkt(Y.iii)
k.a.ii.i ← g.ii.fkt(Y.i)
k.a.ii.ii ← g.ii.fkt(Y.ii)
k.a.ii.iii ← g.ii.fkt(Y.iii)
k.a.iii.i ← g.iii.fkt(Y.i)
k.a.iii.ii ← g.iii.fkt(Y.ii)
k.a.iii.iii ← g.iii.fkt(Y.iii)

k.b.i.i ← h.i.fkt(Y.i)
k.b.i.ii ← h.i.fkt(Y.ii)
k.b.i.iii ← h.i.fkt(Y.iii)
k.b.ii.i ← h.ii.fkt(Y.i)
k.b.ii.ii ← h.ii.fkt(Y.ii)
k.b.ii.iii ← h.ii.fkt(Y.iii)
k.b.iii.i ← h.iii.fkt(Y.i)
```


```
k.b.iii.ii ← h.iii.fkt(Y.ii)
k.b.iii.iii ← h.iii.fkt(Y.iii)


##################################################
# Teil (e)
##################################################

emp.mse ← as.data.frame(matrix(
 c(mean((Y.i - k.a.i.i)^2),
     mean((Y.ii - k.a.i.ii)^2),
     mean((Y.iii - k.a.i.iii)^2),
   mean((Y.i - k.a.ii.i)^2),
     mean((Y.ii - k.a.ii.ii)^2),
     mean((Y.iii - k.a.ii.iii)^2),
   mean((Y.i - k.a.iii.i)^2),
     mean((Y.ii - k.a.iii.ii)^2),
```


```
    mean((Y.iii - k.a.iii.iii)^2),
  mean((Y.i - k.b.i.i)^2),
    mean((Y.ii - k.b.i.ii)^2),
    mean((Y.iii - k.b.i.iii)^2),
  mean((Y.i - k.b.ii.i)^2),
    mean((Y.ii - k.b.ii.ii)^2),
    mean((Y.iii - k.b.ii.iii)^2),
  mean((Y.i - k.b.iii.i)^2),
    mean((Y.ii - k.b.iii.ii)^2),
    mean((Y.iii - k.b.iii.iii)^2)),
  ncol=3, byrow=T,
  dimnames=list(c("(a),i", "(a),ii", "(a),iii",
                  "(b),i", "(b),ii", "(b),iii"),
                   c("normal", "konvex", "cauchy"))
))
```


L.13 Lösungsvorschläge Blatt 13

L.13.1 Lineare Regression

```
##########################################################
# Loesungsvorschlag zu Blatt 13 Aufgabe 1
##########################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME ← "crabs.data"
```


```
crabs ← read.table(file = FILENAME, header = TRUE)
attach(crabs)
names(crabs)
plot(postsz, presz, type = "p")

# Vermutung:
# presz = const + theta*postsz + Fehler
# R-Formel
# postsz ~ presz
```


L.13.2 Freier Fall eines Körpers

```
##################################################
# Loesungsvorschlag zu Blatt 13 Aufgabe 2
##################################################

# Zeitpunkte
t <- seq(from=1/30, to=14/30, by=1/30)
t

# Höhe
h <- c(11.86, 15.67, 20.6, 26.69, 33.71,
       41.93, 51.13, 61.49,72.90, 85.44,
       99.08, 113.77, 129.54, 146.48)
h

# Designmatrix
```


```
eins ← rep(1, length(h))
X ← matrix(c(eins, t, t^2), ncol=3)
X

# Kleinste Quadrate Schätzer
beta.ls ← solve(t(X)%*%X)%*%t(X)%*%h
beta.ls

# Schätzung für Gravitationskonstante g
g ← 2*beta.ls[3]/100
g

# Streung von beta_1
Varianz ← sum((h - beta.ls[1] - beta.ls[2]*t -
            beta.ls[3]*t^2)^2)/(length(h)-3)
Varianz
Kovarianz ← solve(t(X)%*%X)*Varianz
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
Kovarianz
Streuung <- 2*sqrt(Kovarianz[3,3])/100
Streuung
```


L.13.3 Länge der alten Meile

```
##################################################
# Loesungsvorschlag zu Blatt 13 Aufgabe 3
##################################################

# Designmatrix für Modell 1
X1 <- matrix(c(1,1,1,1,1,1,
               1,2,3,4,5,6), ncol=2)
X1

# Beobachtungsvektor
y <- c(784, 2460, 4147, 5826, 7515, 9187)
y

# Kleinste Quadrate Schätzer
beta.ls1 <- solve(t(X1)%*%X1)%*%t(X1)%*%y
```


```
beta.ls1

# Streung von beta_1
Varianz1 <- sum((y - beta.ls1[1] -
          c(1:6)*beta.ls1[2])^2)/(6-2)
Varianz1
Kovarianz1 <- solve(t(X1)%*%X1)*Varianz1
Kovarianz1
Streuung1 <- sqrt(Kovarianz1[2,2])
Streuung1



# Designmatrix für Modell 2
X2 <- matrix(rep(1,5), ncol=1)
X2

# Beobachtungsvektor
```


```
z ← y[2:6]-y[1:5]
z

# Kleinste Quadrate Schätzer
beta.ls2 ← solve(t(X2)%*%X2)%*%t(X2)%*%z
beta.ls2

# Streung von beta_1
Varianz2 ← sum((z - beta.ls2)^2)/(5-1)
Varianz2
Kovarianz2 ← solve(t(X2)%*%X2)*Varianz2
Kovarianz2
Streuung2 ← sqrt(Kovarianz2)
Streuung2
```


L.13.4 Modellanpassung, Modellwahl

```
##################################################
# Loesungsvorschlag zu Blatt 13 Aufgabe 4
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "model.txt"

model <- read.table(file = FILENAME, header = TRUE)
attach(model)
```


model

```
#############################################################
# Teil (a)
#############################################################
# backward selection
lm.iii <- lm(Y ~ X + I(X^2))
summary(lm.iii)
lm.ii <- update(lm.iii, .~. - I(X^2))
summary(lm.ii)

# forward selection
lm.i <- lm(Y ~ 1)
summary(lm.i)
lm.ii <- update(lm.i, .~. + X)
summary(lm.ii)
lm.iii <- update(lm.ii, .~. + I(X^2))
```


```
summary(lm.iii)

# AIC Kriterium
step(lm.iii)
drop1(lm.iii, test="F")
drop1(lm.iii, test="Chisq")

# Cp Kriterium
require(leaps)
lm1 ← leaps(matrix(c(X, X^2), ncol=2), Y)
# nehme das Modell mit cp=p also hier y=ax+b

##################################################
# Teil (b)
##################################################
plot(X, Y)
lines(X, fitted(lm.ii))
```


```
title("Plot der angepassten Kurve")

##################################################
# Teil (c)
##################################################
# 1. Moeglichkeit
#kurve <- splinefun(X, fitted(lm.ii))
#points(-2, kurve(-2), col="red")
# 2. Moeglichkeit
neu1 <- data.frame(X = -2)
points(-2, predict(lm.ii, neu1, se.fit=T)$fit,
       col="red")

neu <- data.frame(X = seq(-3, -1, 0.1))
pred.KI <- predict(lm.ii, neu, interval="prediction")
lines(neu$X, pred.KI[,2], col="red")
lines(neu$X, pred.KI[,3], col="red")
```


```
##################################################
# Teil (d)
##################################################
FILENAME ← "luecke.txt"

luecke ← read.table(file = FILENAME, header = TRUE)
points(luecke[,2], luecke[,1], col="blue")
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

L.14 Lösungsvorschläge Blatt 14

L.14.1 ANOVA

```
#############################################################
# Loesungsvorschlag zu Blatt 14 Aufgabe 1
#############################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME ← "margarine.txt"
```


```
margarine ← read.table(file = FILENAME, header = TRUE)
margarine

# Kovariate als Faktoren kodieren
margarine[,2] ← factor(margarine[,2])
margarine[,3] ← factor(margarine[,3])
attach(margarine)

par(mfrow=c(2,1))
plot(Y ~ FaktorA * FaktorB)
windows()
par(mfrow=c(2,1))
interaction.plot(FaktorA, FaktorB, Y)
interaction.plot(FaktorB, FaktorA, Y)
# Geraden nicht parallel - es gibt Interaktionen
lm1 ← lm(Y ~ FaktorA * FaktorB)
```


```
print(lm1)
summary(lm1)
plot(lm1)
anova(lm1)

windows()
par(mfrow=c(2,1))
qqnorm(lm1$res)
qqline(lm1$res)
plot(lm1$fitted, lm1$res, xlab="gefittete Werte",
     ylab="Residuen")
title("Residuenplot")
```


L.14.3 Box–Cox–Transformation I

```
##################################################
# Loesungsvorschlag zu Blatt 14 Aufgabe 3
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME ← "gauge.txt"

gauge ← ← read.table(file = FILENAME, header = TRUE)
```


```
attach(gauge)
gauge

##################################################
# Teil (a)
##################################################
lm1 ← lm(gain ~ density)
summary(lm1)
par(mfrow=c(2,1))
plot(density, lm1$resid, xlab="density",
     ylab="Residuen")
title(main="Residuenplots für die Originaldaten")
plot(lm1$fitted, lm1$resid, xlab="gefittete Werte",
     ylab="Residuen")
# quadratische Funktion in density...

##################################################
```


```
# Teil (b)
##########################################################
require(MASS)

# 1. Versuch: lambda=default=seq(-3, 3, 0.1)
windows()
par(mfrow=c(1,3))
boxcox(object=gain ~ density)

# 2. Versuch:
boxcox(object=gain ~ density,
       lambda=seq(-0.2, 0, 0.01))

# 3. Versuch:
boxcox(object=gain ~ density,
       lambda=seq(-0.1, -0.05, 0.01))
```


```
gain1 ← gain^(-0.09)

##############################################
# Teil (c)
##############################################
lm2 ← lm(gain1 ~ density)
summary(lm2)
windows()
par(mfrow=c(2,1))
plot(density, resid(lm2), xlab="density",
     ylab="residuen(gain)")
title("Residuenplot für die transformierten Daten")
plot(lm2$fitted, lm2$resid, xlab="gefittete Werte",
     ylab="Residuen")

##############################################
# Teil (d)
```


```
##################################################
windows()
plot(density, gain)
lines(density, fitted(lm1), col="blue")
lines(density, fitted(lm2)^(-1/0.09), col="red")
title("Der gauge Datensatz mit angepassten Kurven")
legend(min(density), 100, legend=c("original",
        "transformiert"), fill=c("blue", "red"))
```


Peter Ruckdeschel
Matthias Kohl

L.14.4 Box–Cox–Transformation II

```
##################################################
# Loesungsvorschlag zu Blatt 14 Aufgabe 4
##################################################

# aus Blatt 12 Aufgabe 3
# E exakt

Eexakt <- function(n){n*sum(1/(1:n))}
n <- 1:100
y <- sapply(n,Eexakt)
yn <- sapply(n,Eexakt)/n

require(MASS)

# ACHTUNG boxcox transformiert die y's nicht die x's
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
# -> Vertauschung von y <-> x

par(mfrow=c(1,2))
boxcox(object=n ~ y) # => ueberlinear
boxcox(object=n ~ yn) # ~ 0  also log
par(mfrow=c(1,1))

# also E(n)= a * log(n)*n + c ?
lm(y~I(log(n)*n))
plot(n, yn)
par(new="T")
plot(n, 1.12*n*log(n)+4.724)
matplot(n, cbind(y, 1.12*n*log(n)+4.724))
res <- y - 1.12*n*log(n) - 4.72
plot(res)
# so noch nicht befriedigend...
```


```
# also E(n)/n= a * log(n) + c ?
lm(yn~I(log(n)))
res21 <- yn - 0.96*log(n) - 0.76
res2 <- n*(yn - 0.96*log(n) - 0.76)
plot(res21)
plot(res2)

# oder : E(n)= a * n* log(n) + b*n + c ?
lm(y~I(n*log(n))+n)
res3 <- lm(yn~I(n*log(n))+n)$res
plot(res3)
# gar nicht schlecht, aber...
# Asymptotik nicht gut fuer kleines n?

erg <- lm(y[10:100]~I(n[10:100]*log(n[10:100]))
          + n[10:100])
print(erg)
```


```
res4 ← y − erg$co[1] − erg$co[2]*n*log(n)
       − n*erg$co[3]
plot(res4)
# schon fast prima...

erg2 ← lm(y[20:100]~I(n[20:100]*log(n[20:100]))
          + n[20:100])
print(erg2)
res5 ← y − erg2$co[1] − erg2$co[2]*n*log(n)
       − n*erg2$co[3]
plot(res5)
plot(erg2$res)

# prima, oder ?
N←1:100000
mach ← sum(1/N) − log(100000)
print(c(mach=mach, gesch=erg2$co[3],
```


```
        diff=mach-erg2$co[3]))
# -> auf 3 Stellen genau!
```


Peter Ruckdeschel
Matthias Kohl

L.14.5 Generalisiert lineares Modell

```
##########################################################
# Loesungsvorschlag zu Blatt 14 Aufgabe 5
##########################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "adver.txt"

adver <- read.table(file = FILENAME, header = TRUE)
attach(adver)
```


```
adver

# erste Spalte: Wochenzaehler,
# zweite Spalte: Anzahl der "JA"
# dritte Spalte: Budget

# erzeugen der Response - Variablen : aus
# technischen Gruenden (MISSINGS!)
# benoetigt R die Zahl der ja's in einer Spalte,
# die Zahl der nein's in einer zweiten
personen <- matrix(c(pers, 66-pers), ncol=2)

glm1 <- glm(formula=personen ~ aufw, family=binomial)
glm2 <- glm(formula=personen ~ aufw,
            family=binomial(link=probit))

summary(glm1)
```


```
summary(glm2)

windows()
par(mfrow=c(2,2))
plot(glm1)
windows()
par(mfrow=c(2,2))
plot(glm2)

# Unterstellter Zusammenhang
x <- seq(0, max(aufw)*1.1, 0.1)
# lineare Prediktoren
lpred1 <- glm1$coe[1] + glm1$coe[2]*x
lpred2 <- glm2$coe[1] + glm2$coe[2]*x
# Link
link1 <- exp(lpred1)/(1 + exp(lpred1))
link2 <- pnorm(lpred2)
```


```
windows()
matplot(x, cbind(link1, link2), type="l",
        ylim=c(-0.05, max(pers/66)+0.05),
        ylab="Prognose fuer die Wirksamkeit",
        xlab="Budget")
points(aufw,pers/66)
detach()
# kaum Unterschiede zwsichen Probit und logit
```


L.15 Lösungsvorschläge Blatt 15

L.15.1 Multivariate Normalverteilung

```
##################################################
# Loesungsvorschlag zu Blatt 15 Aufgabe 1
##################################################

##################################################
# Teil (a)
##################################################
# zieht die Wurzel aus einer positiv definiten
# Matrix mit svd
wurzel <- function(x){# Diagnostik:
    if(nrow(x) != ncol(x))
        stop("keine quadratische Matrix!")
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
    if(sum(t(x) != x) > 0)
        stop("keine symmetrische Matrix!")
    erg <- svd(x)
    if(sum(eigen(x, symmetric = TRUE)$values < 0)
        > 0)
        stop("keine pos. semidefinite Matrix!")

    return(erg$u %*% diag(sqrt(erg$d)) %*% t(erg$v))
}

# zieht die Wurzel aus einer positiv definiten
# Matrix mit choleski
wurzel2 <- function(x){# Diagnostik:
    if(nrow(x) != ncol(x))
        stop("keine quadratische Matrix!")
    if(sum(t(x) != x) > 0)
        stop("keine symmetrische Matrix!")
```


```
    erg <- t(chol(x))

    return(erg)
}

# mehrdimensionale Normalverteilung
rnnorm <- function(n, mu, sigma){
    d <- length(mu)
    X0 <- matrix(rnorm(d*n), d, n)
    X <- t(wurzel(sigma)%*%X0 + mu)

    return(X)
}

##################################################
# Teil (b)
##################################################
```


```
# mehrdimensionale Konvexkombination von
# Normalverteilungen
contnorm ← function(n, mu.id, sigma.id, mu.c,
                    sigma.c, r){
    X.id ← rnnorm(n, mu.id, sigma.id)
    X.c ← rnnorm(n, mu.c, sigma.c)
    U ← rbinom(n, 1, r)
    X ← (1−U)∗X.id + U∗X.c

    return(list(r = X, id = X.id, c = X.c, U = U))
}
S.id ← matrix(c(1, −1, −1, 2), 2, 2)
m.id ← c(0, 0)
S.c ← matrix(c(0.1, 0.15, 0.15, 4), 2, 2)
m.c ← c(5, −2)
X ← contnorm(n = 200, mu.id = m.id, sigma.id = S.id,
        mu.c = m.c, sigma.c = S.c, r = 0.1)
```


```
##################################################
# Teil (c)
##################################################
rad <- qnorm(1 - 0.025)
zerl <- svd(S.id)
theta <- seq(from = 0, to = 2*pi, by = 0.01)
ellipse <- rbind(rad*cos(theta)*sqrt(zerl$d[1]),
                 rad*sin(theta)*sqrt(zerl$d[2]))
ellipse <- t(zerl$u %*% ellipse)

#Grenzen für die Plots
re <- ceiling(max(ellipse[ ,1], X$r[ ,1],
                  X$id[ ,1])+0.5)
li <- floor(min(ellipse[,1], X$r[,1],
                X$id[ ,1])-0.5)
ob <- ceiling(max(ellipse[,2], X$r[ ,2],
```


```
                X$id[ ,2])+0.5)
un ← floor(min(ellipse[,2], X$r[ ,2],
             X$id[ ,2])-0.5)

par(mfrow = c(1, 1), col = "dark red")
plot(ellipse, type = "l", xlab = "X1", ylab = "X2",
     xlim = c(li,re), ylim = c(un,ob), lwd = 2)

# feststellen ob in oder nicht in Ellipse
##################################################
# Teil (d)
##################################################
X.im.u ← t(zerl$u %*% t(X$r))
ind.in.ell ← ((X.im.u[,1]^2/zerl$d[1] +
     X.im.u[,2]^2/zerl$d[2]) <= rad^2)
# innerhalb der Ellipse und ideal
X.im.id ← X$r[ind.in.ell&(X$U == 0), ]
```


```
# innerhalb der Ellipse und cont
X.im.c ← X$r[ind.in.ell&(X$U == 1), ]
# ausserhalb der Ellipse und ideal
X.aus.id ← X$r[(!ind.in.ell)&(X$U == 0), ]
# ausserhalb der Ellipse und cont
X.aus.c n ← X$r[(!ind.in.ell)&(X$U == 1), ]

par(new = TRUE, col = "green")
plot(X.im.id, type = "p", xlab = "X1", ylab = "X2",
        xlim = c(li,re), ylim = c(un,ob))
par(new = TRUE, col = "blue")
plot(X.aus.id, type = "p", xlab = "X1", ylab = "X2",
        xlim = c(li,re), ylim = c(un,ob))
par(new = TRUE, col = "red")
plot(X.in.c, Xbcont.i2, type = "p", xlab = "X1",
    ylab = "X2", xlim = c(li,re), ylim = c(un,ob))
par(new = TRUE, col = "orange")
```


```
plot(X.aus.c, type = "p", xlab = "X1",
     ylab = "X2", xlim = c(li,re), ylim = c(un,ob))
par(new = TRUE, col = 1)
plot(0, 0, type = "n", xlab = "X1", ylab = "X2",
     xlim = c(li,re), ylim = c(un,ob))
title(expression(paste("Bivariate Konvexkombination ",
       (1-r), italic(N), "(", mu[id], ",", Sigma[id],
       ") ", + r, italic(N), "(", mu[cont], ",",
       Sigma[cont], ")")))
text(0.5, 3, "95% Konfidenzintervall",
       col = "dark red", cex = 1.1)
legend(li, un+4, legend = c("ideal innerhalb",
       "ideal außerhalb", "kont innerhalb",
       "kont außerhalb"), pch = 1,
       col = c("green", "blue", "red", "orange"))
legend(li, un+4, legend = c("ideal innerhalb",
       "ideal außerhalb", "kont innerhalb",
```


```
"kont außerhalb"), pch = 1,
col = c("green", "blue", "red" "orange"))
```


L.15.2 Clustering, Diskriminanzanalyse

```
##################################################
# Loesungsvorschlag zu Blatt 15 Aufgabe 2
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "bank2.txt"
bank2 <- read.table(file = FILENAME, header = FALSE)

##################################################
```


```r
# Teil (a)
##########################################################
require(mva)
# complete linkage
h1 <- hclust(dist(bank2), method = "complete")
plot(h1, main = "Complete Linkage")

# single linkage
h2 <- hclust(dist(bank2), method = "single")
windows()
plot(h2, main = "Single Linkage")

# average linkage
h3 <- hclust(dist(bank2), method = "average")
windows()
plot(h3, main = "Average Linkage")
```


```
# Ward-Algorithmus
h4 <- hclust(dist(bank2), method = "ward")
windows()
plot(h4, main = "Ward-Algorithmus")

# Centroid-Algorithmus
h5 <- hclust(dist(bank2), method = "centroid")
windows()
plot(h5, main = "Centroid-Algorithmus")

# McQuitty-Algorithmus
h6 <- hclust(dist(bank2), method = "mcquitty")
windows()
plot(h6, main = "McQuitty-Algorithmus")

# Median-Algorithmus
h7 <- hclust(dist(bank2), method = "median")
```


```
windows()
plot(h7, main = "Median-Algorithmus")

##################################################
# Teil (b)
##################################################
#e = echte und f = falsche Banknoten
groups <- c(rep("e",100),rep("f",100))
bank <- cbind(bank2,groups)

# Trainings und Validierungsstichprobe
train <- c(1:70, 131:200)
valid <- c(71:130)

require(MASS)
# Lineare DA
l <- lda(groups ~ ., bank, subset = train)
```


```
l.pre ← predict(l, bank[valid,])

# Quadratische DA
q ← qda(groups ~ ., bank, subset = train)
q.pre ← predict(q, bank[valid,])

l.pre$class
q.pre$class
```


L.15.3 Hauptkomponentenanalyse, Faktoranalyse

```
##################################################
# Loesungsvorschlag zu Blatt 15 Aufgabe 3
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "bank2.txt"
bank2 <- read.table(file = FILENAME, header = FALSE)

##################################################
```


```
# Teil(a)
##########################################################
# 1. Moeglichkeit
bank.pr <- prcomp(scale(bank2, center = TRUE,
                        scale = TRUE))
bank.pr
summary(bank.pr)

bank.pr1 <- prcomp(bank2, center = TRUE)
bank.pr1
summary(bank.pr1)
# equal scaled plot
eqscplot(bank.pr$x[,1:2], type = "n",
         xlab = "1. Hauptkomp.",
         ylab = "2. Hauptkomp.")
text(bank.pr$x[,1:2],
     c(rep("e",100),rep("f",100)))
```


```
title("Hauptkomponentenanalyse für Banknoten")
windows()
eqscplot(bank.pr1$x[,1:2], type = "n",
                xlab = "1. Hauptkomp.",
                ylab = "2. Hauptkomp.")
text(bank.pr1$x[,1:2],
     c(rep("e",100), rep("f",100)))
title("Hauptkomponentenanalyse für Banknoten")

# 2. Moeglichkeit
bank.prin ← princomp(bank2)
summary(bank.prin)
plot(bank.prin)
biplot(bank.prin)

bank.prin1 ← princomp(bank2, cor = TRUE)
summary(bank.prin1)
```


```
plot(bank.prin1)
biplot(bank.prin1)

##################################################
# Teil(b)
##################################################
# 2 Faktoren
fact2 <- factanal(bank2, factors = 2)
fact2
loadings(fact2)
varimax(fact2$loadings, normalize = FALSE)
varimax(fact2$loadings)
promax(fact2$loadings)

# 3 Faktoren
fact3 <- factanal(bank2, factors = 3)
fact3
```


```
loadings(fact3)
varimax(fact3$loadings, normalize = FALSE)
varimax(fact2$loadings)
promax(fact3$loadings)
```


L.15.5 normalisierte Hauptkomponentenanalyse

```
#####################################################
# Loesungsvorschlag zu Blatt 15 Aufgabe 5
#####################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "uscrime.dat"
uscrime <- read.table(file = FILENAME, header = FALSE)
```


```
##########################################################
# Teil (a)
##########################################################

crime0 <- as.matrix(uscrime[,c(1:9)])

#standardisierung

crimem<-apply(crime0,2,mean)
crimes<-apply(crime0,2,sd)

crime<-t((t(crime0)-crimem)/crimes)

R <- cor(crime)

# Eigenwerte und Eigenvektoren der
# Korrelationsmatrix
```


```
EWR <- eigen(R)$values
EVktR <- eigen(R)$vectors
EWR
EVktR

#Erklärte Varianz
psi1 <- EWR[1]/sum(EWR)
psi2 <- (EWR[1]+EWR[2])/sum(EWR)
psi3 <- (EWR[1]+EWR[2]+EWR[3])/sum(EWR)
psi1
psi2
psi3

#Unterschiede zwischen den Regionen
#Berechnung der NPCs
NPC1 <- crime %*% EVktR[,1]
NPC2 <- crime %*% EVktR[,2]
```


```
NPC3 ← crime %*% EVktR[,3]

ind1←uscrime[,10] == 1
ind2←uscrime[,10] == 2
ind3←uscrime[,10] == 3
ind4←uscrime[,10] == 4

#Plots
par(mfrow = c(2,2))
plot(NPC1, NPC2, type = "p", xlab = "NPC1",
      ylab = "NPC2", main = "erste␣gegen␣zweite␣NPC")
text(NPC1[ind1 == 1], NPC2[ind1 == 1],
      (uscrime[,10])[ind1 == 1],col = "red")
text(NPC1[ind2 == 1], NPC2[ind2 == 1],
      (uscrime[,10])[ind2 == 1],col = "green")
text(NPC1[ind3 == 1], NPC2[ind3 == 1],
      (uscrime[,10])[ind3 == 1],col = "black")
```


```
text(NPC1[ind4 == 1], NPC2[ind4 == 1],
     (uscrime[,10])[ind4 == 1],col = "blue")

plot(NPC2, NPC3, type = "p", xlab = "NPC2",
     ylab = "NPC3", main = "zweite gegen dritte NPC")
text(NPC2[ind1 == 1], NPC3[ind1 == 1],
     (uscrime[,10])[ind1 == 1],col = "red")
text(NPC2[ind2 == 1], NPC3[ind2 == 1],
     (uscrime[,10])[ind2 == 1],col = "green")
text(NPC2[ind3 == 1], NPC3[ind3 == 1],
     (uscrime[,10])[ind3 == 1],col = "black")
text(NPC2[ind4 == 1], NPC3[ind4 == 1],
     (uscrime[,10])[ind4 == 1],col = "blue")

plot(NPC1, NPC3, type = "p", xlab = "NPC1",
     ylab = "NPC3", main = "erste gegen dritte NPC")
text(NPC1[ind1 == 1], NPC3[ind1 == 1],
```


```
      (uscrime[,10])[ind1 == 1], col = "red")
text(NPC1[ind2 == 1], NPC3[ind2 == 1],
      (uscrime[,10])[ind2 == 1], col = "green")
text(NPC1[ind3 == 1], NPC3[ind3 == 1],
      (uscrime[,10])[ind3 == 1], col = "black")
text(NPC1[ind4 == 1], NPC3[ind4 == 1],
      (uscrime[,10])[ind4 == 1], col = "blue")

plot(c(1:9), EWR, type = "p", xlab = "Index",
      ylab = "Lambda", main = "Eigenwerte")

##################################################
# Teil (b)
##################################################
#Ohne Variable area of state
crimeb ← crime[,2:9]
```


```
Rb ← cor(crimeb)

# Eigenwerte und Eigenvektoren der
# Korrelationsmatrix
EWRb ← eigen(Rb)$values
EVktRb ← eigen(Rb)$vectors
EWRb
EVktRb

#Erklärte Varianz
psi1b ← EWRb[1]/sum(EWRb)
psi2b ← (EWRb[1]+EWRb[2])/sum(EWRb)
psi3b ← (EWRb[1]+EWRb[2]+EWRb[3])/sum(EWRb)
psi1b
psi2b
psi3b
```


```
#Unterschiede zwischen den Regionen
#Berechnung der NPCs
NPC1b ← crimeb %*% EVktRb[,1]
NPC2b ← crimeb %*% EVktRb[,2]
NPC3b ← crimeb %*% EVktRb[,3]

#Plots
par(mfrow = c(2,2))
plot(NPC1b, NPC2b, type = "p", xlab = "NPCb1",
     ylab = "NPC2b", main = "erste gegen zweite NPC - b")
text(NPC1b[ind1 == 1], NPC2b[ind1 == 1],
     (uscrime[,10])[ind1 == 1],col = "red")
text(NPC1b[ind2 == 1], NPC2b[ind2 == 1],
     (uscrime[,10])[ind2 == 1],col = "green")
text(NPC1b[ind3 == 1], NPC2b[ind3 == 1],
     (uscrime[,10])[ind3 == 1],col = "black")
text(NPC1b[ind4 == 1], NPC2b[ind4 == 1],
```


```
      (uscrime[,10])[ind4 == 1],col = "blue")

plot(NPC2b, NPC3b, type = "p", xlab = "NPC2",
      ylab = "NPC3", main = "zweite gegen dritte NPC - b")
text(NPC2b[ind1 == 1], NPC3b[ind1 == 1],
      (uscrime[,10])[ind1 == 1],col = "red")
text(NPC2b[ind2 == 1], NPC3b[ind2 == 1],
      (uscrime[,10])[ind2 == 1],col = "green")
text(NPC2b[ind3 == 1], NPC3b[ind3 == 1],
      (uscrime[,10])[ind3 == 1],col = "black")
text(NPC2b[ind4 == 1], NPC3b[ind4 == 1],
      (uscrime[,10])[ind4 == 1],col = "blue")

plot(NPC1b, NPC3b, type="p", xlab="NPC1",
      ylab = "NPC3", main = "erste gegen dritte NPC - b")
text(NPC1b[ind1 == 1], NPC3b[ind1 == 1],
      (uscrime[,10])[ind1 == 1],col = "red")
```


```
text(NPC1b[ind2 == 1], NPC3b[ind2 == 1],
     (uscrime[,10])[ind2 == 1],col = "green")
text(NPC1b[ind3 == 1], NPC3b[ind3 == 1],
     (uscrime[,10])[ind3 == 1],col = "black")
text(NPC1b[ind4 == 1], NPC3b[ind4 == 1],
     (uscrime[,10])[ind4 == 1],col = "blue")

plot(c(1:8), EWRb, type = "p", xlab = "Index",
     ylab = "Lambda", main = "Eigenwerte")
```


L.16 Lösungsvorschläge Blatt 16

L.16.1 Erstellung von Zeitreihenobjekten

```
##########################################################
# Loesungsvorschlag zu Blatt 16 Aufgabe 1
##########################################################

require(tseries)
require(its)
require(zoo)
require(chron)

#(a)
IBM.ts <- get.hist.quote(instrument = "ibm",
                         start = "1998-01-01",
```

UNIVERSITÄT BAYREUTH
Mathematik VII
Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
                    end = "2006-07-11",
                    quote = "Close",
                    retclass = "ts")


tsd <- dates("1998-01-01",format="y-m-d")+1:length(IBM.ts)

#(b)
dates0=as.POSIXct(x = strptime(tsd, format = "%y-%m-%d"))
IBM.its <- its(x = IBM.ts[1:3112], dates = dates0[1:3112],
               format = its.format("%d. %B %Y"))
(IBM.its <- IBM.its [!is.na(IBM.its)])
##oder
IBM.its <- get.hist.quote(instrument = "ibm",
                    start = "1998-01-01",
                    end = "2006-07-11",
                    quote = "Close",
```


```
                    retclass = "its")

its.format("%d.%B.%Y")

row.names(IBM.its)


plot(IBM.its, main = "International Business Machines Corp")
#(b)
IBM.zoo ← as.zoo(IBM.its)
##oder
IBM.zoo ← get.hist.quote(instrument = "ibm",
                         start = "1998-01-01",
                         end = "2006-07-11",
                         quote = "Close",
                         retclass = "zoo")
```


```
rm(list=ls(all=TRUE))
```


L.16.2 Zeitreihenanalyse I

```
##################################################
# Loesungsvorschlag zu Blatt 16 Aufgabe 2
##################################################

## Vorarbeit:
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME <- "daten01.txt"
zeitd <- read.table(file = FILENAME, header = TRUE)
```


```
##############################################################
# Teil (a)
##############################################################

par(mfrow=c(4,3))
plot(zeitd$"V1",zeitd$"V1",type="l")
plot(zeitd$"V1",zeitd$"V2",type="l")
plot(zeitd$"V1",zeitd$"V3",type="l")
plot(zeitd$"V1",zeitd$"V4",type="l")
plot(zeitd$"V1",zeitd$"V5",type="l")
plot(zeitd$"V1",zeitd$"V6",type="l")
plot(zeitd$"V1",zeitd$"V7",type="l")
plot(zeitd$"V1",zeitd$"V8",type="l")
plot(zeitd$"V1",zeitd$"V9",type="l")
plot(zeitd$"V1",zeitd$"V10",type="l")
plot(zeitd$"V1",zeitd$"V11",type="l")
plot(zeitd$"V1",zeitd$"V12",type="l")
```


```
par(mfrow=c(1,1))

#"stat": 2 4 8 9 10 11

## ab jetzt in dieser Reihenfolge

par(mfrow=c(4,3))
plot(zeitd$"V1",zeitd$"V1",type="l")
plot(zeitd$"V1",zeitd$"V3",type="l")
plot(zeitd$"V1",zeitd$"V5",type="l")
plot(zeitd$"V1",zeitd$"V6",type="l")
plot(zeitd$"V1",zeitd$"V7",type="l")
plot(zeitd$"V1",zeitd$"V12",type="l")
plot(zeitd$"V1",zeitd$"V2",type="l")
plot(zeitd$"V1",zeitd$"V4",type="l")
plot(zeitd$"V1",zeitd$"V8",type="l")
plot(zeitd$"V1",zeitd$"V9",type="l")
```


```
plot(zeitd$"V1",zeitd$"V10",type="l")
plot(zeitd$"V1",zeitd$"V11",type="l")
par(mfrow=c(1,1))



##################################################
# Teil (b)
##################################################


par(mfrow=c(4,3))
acf(zeitd$"V1",type="covariance")
acf(zeitd$"V3",type="covariance")
acf(zeitd$"V5",type="covariance")
acf(zeitd$"V6",type="covariance")
acf(zeitd$"V7",type="covariance")
acf(zeitd$"V12",type="covariance")
acf(zeitd$"V2",type="covariance")
```


```
acf(zeitd$"V4",type="covariance")
acf(zeitd$"V8",type="covariance")
acf(zeitd$"V9",type="covariance")
acf(zeitd$"V10",type="covariance")
acf(zeitd$"V11",type="covariance")
par(mfrow=c(1,1))

par(mfrow=c(4,3))
acf(zeitd$"V1",type="correlation")
acf(zeitd$"V3",type="correlation")
acf(zeitd$"V5",type="correlation")
acf(zeitd$"V6",type="correlation")
acf(zeitd$"V7",type="correlation")
acf(zeitd$"V12",type="correlation")
par(mfrow=c(1,1))

par(mfrow=c(4,3))
```


```
acf(zeitd$"V1",type="partial")
acf(zeitd$"V3",type="partial")
acf(zeitd$"V5",type="partial")
acf(zeitd$"V6",type="partial")
acf(zeitd$"V7",type="partial")
acf(zeitd$"V12",type="partial")
acf(zeitd$"V2",type="partial")
acf(zeitd$"V4",type="partial")
acf(zeitd$"V8",type="partial")
acf(zeitd$"V9",type="partial")
acf(zeitd$"V10",type="partial")
acf(zeitd$"V11",type="partial")
par(mfrow=c(1,1))

## nur die stationaeren

par(mfrow=c(4,3))
```


```
acf(zeitd$"V2",type="correlation")
acf(zeitd$"V4",type="correlation")
acf(zeitd$"V8",type="correlation")
acf(zeitd$"V9",type="correlation")
acf(zeitd$"V10",type="correlation")
acf(zeitd$"V11",type="correlation")
acf(zeitd$"V2",type="partial")
acf(zeitd$"V4",type="partial")
acf(zeitd$"V8",type="partial")
acf(zeitd$"V9",type="partial")
acf(zeitd$"V10",type="partial")
acf(zeitd$"V11",type="partial")
par(mfrow=c(1,1))

#=>
#2 weisses Rauschen?
#4 AR(1) mit pos. Koeff?
```


```
#8 weisses Rauschen?
#9 AR(1) mit pos. Koeff?
#10 MA(1) mit pos. Koeff?
#11 ARMA(1,1) oder (2,2)?


##################################################
# Teil (c)
##################################################

W1<- diff(zeitd$"V1",1,1)
W3<- diff(zeitd$"V3",1,1)
W5<- diff(zeitd$"V5",1,1)
W6<- diff(zeitd$"V6",1,1)
W7<- diff(zeitd$"V7",1,1)
W12<- diff(zeitd$"V12",1,1)
```


```
WW1<- diff(zeitd$"V1",1,2)
WW3<- diff(zeitd$"V3",1,2)
WW5<- diff(zeitd$"V5",1,2)
WW6<- diff(zeitd$"V6",1,2)
WW7<- diff(zeitd$"V7",1,2)
WW12<- diff(zeitd$"V12",1,2)

zeitdd <- list(W1=W1,W3=W3,W5=W5,W6=W6,
               W7=W7,W12=W12,
               WW1=WW1,WW3=WW3,WW5=WW5,
               WW6=WW6,WW7=WW7,WW12=WW12)

par(mfrow=c(4,3))
plot(2:100,zeitdd$"W1",type="l")
plot(2:100,zeitdd$"W3",type="l")
plot(2:100,zeitdd$"W5",type="l")
```


```
plot(2:100,zeitdd$"W6",type="l")
plot(2:100,zeitdd$"W7",type="l")
plot(2:100,zeitdd$"W12",type="l")
plot(3:100,zeitdd$"WW1",type="l")
plot(3:100,zeitdd$"WW3",type="l")
plot(3:100,zeitdd$"WW5",type="l")
plot(3:100,zeitdd$"WW6",type="l")
plot(3:100,zeitdd$"WW7",type="l")
plot(3:100,zeitdd$"WW12",type="l")
par(mfrow=c(1,1))
```


L.16.3 Zeitreihenanalyse II

```
##################################################
# Loesungsvorschlag zu Blatt 16 Aufgabe 3
##################################################

##################################################
# Teil (a)
##################################################
require(ts)

X <- arima.sim(n=100, list(ar=.7, ma=.35),
               sd=sqrt(0.5))
X
plot(X)
ac1 <- acf(X, type="correlation")
pac1 <- pacf(X)
```


Peter Ruckdeschel
Matthias Kohl

R/S-plus für Einsteiger und Fortgeschrittene

```
est ← arima(X, order=c(1,0,1))
est
plot(est$residuals)

##################################################
# Teil (b)
##################################################
data(LakeHuron)
LH2 ← LakeHuron - 570
LH2z ← LH2 - mean(LH2)
plot(LH2z)
est2 ← arima(LH2, order=c(1,0,1))
est2
plot(est2$residuals)
ac2 ← acf(LH2, type="correlation")
pac2 ← pacf(LH2)
```


```
# Vorhersage
predict(est2, n.ahead = 5)
prog <- predict(est2, n.ahead = 5)$pred
            + mean(LH2) + 570
```


L.16.4 Räumliche Statistik

```
##################################################
# Loesungsvorschlag zu Blatt 16 Aufgabe 5
##################################################

# Nach Skript file geoRintro.R
##
## ------------------------------------------------
## Commands included in the geoRintro web-page
## ------------------------------------------------
##
## 1. Sourcing the package
##
require(geoR)

## Vorarbeit:
```


```
## setzte das Directory:
#  setwd("<Ihr Pfad>")
## zB
#  setwd("C:/eigene Dateien/Arbeit/Uebungen/R/TeX/")
####

FILENAME ← "s101.RData"
load(FILENAME)
par.ori ← par(no.readonly=TRUE)
##
## 2. Descriptive plots
##
plot(s101)
##
par(mfrow = c(2,2), mar=c(3,3,1,1), mgp=c(2,1,0))
points(s101, xlab = "Coord X", ylab = "Coord Y")
points(s101, xlab = "Coord X", ylab = "Coord Y",
```


```
        pt.divide = "rank.prop")
points(s101, xlab = "Coord X", ylab = "Coord Y",
        cex.max = 1.7,
        col = gray(seq(1, 0.1, l=100)),
        pt.divide = "equal")
points(s101, pt.divide = "quintile",
        xlab = "Coord X", ylab = "Coord Y")
par(par.ori)
##
## 3. Variograms
##
cloud1 ← variog(s101, option = "cloud", max.dist=1)
cloud2 ← variog(s101, option = "cloud",
            estimator.type = "modulus", max.dist=1)
bin1 ← variog(s101, uvec=seq(0,1,l=11))
bin2 ← variog(s101, uvec=seq(0,1,l=11),
            estimator.type= "modulus")
```


```
par(mfrow=c(2,2))
plot(cloud1, main = "classical estimator")
plot(cloud2, main = "modulus estimator")
plot(bin1, main = "classical estimator")
plot(bin2, main = "modulus estimator")
par(par.ori)
##
bin1 ← variog(s101,uvec = seq(0,1,l=11),
              bin.cloud = T)
bin2 ← variog(s101,uvec = seq(0,1,l=11),
              estimator.type = "modulus",
              bin.cloud = T)
par(mfrow = c(1,2))
plot(bin1, bin.cloud = T,
     main = "classical estimator")
plot(bin2, bin.cloud = T,
     main = "modulus estimator")
```


```
par(par.ori)
##
bin1 ← variog(s101, uvec = seq(0,1,l=11))
plot(bin1)
lines.variomodel(list(nugget = 0,
        cov.pars = c(1,0.3), max.dist = 1,
        cov.model = "exp"), lwd = 3)
smooth ← variog(s101, option = "smooth",
            max.dist = 1, n.points = 100,
            kernel = "normal", band = 0.2)
lines(smooth, type ="l", lty = 2)
legend(0.3, 0.4,
    c("empirical", "exponential␣model", "smoothed"),
    lty = c(1,1,2), lwd = c(1,3,1))
##
vario60 ← variog(s101, max.dist = 1, direction=pi/3)
vario.4 ← variog4(s101, max.dist = 1)
```


```
plot(vario.4)
##
## 4. Parameter estimation
##
## Fitting models with nugget fixed to zero
ml <- likfit(s101, ini = c(1,0.5), fix.nugget = T)
reml <- likfit(s101, ini = c(1,0.5), fix.nugget = T,
               method = "RML")
ols <- variofit(bin1, ini = c(1,0.5), fix.nugget = T,
               weights="equal")
wls <- variofit(bin1, ini = c(1,0.5), fix.nugget = T)
## Fitting models with a fixed value for the nugget
ml.fn <- likfit(s101, ini = c(1,0.5), fix.nugget = T,
               nugget = 0.15)
reml.fn <- likfit(s101, ini = c(1,0.5),
                  fix.nugget = T, nugget = 0.15,
                  method = "RML")
```


```
ols.fn ← variofit(bin1,ini = c(1,0.5),
                  fix.nugget = T, nugget = 0.15,
                  weights="equal")
wls.fn ← variofit(bin1, ini = c(1,0.5),
                  fix.nugget = T, nugget = 0.15)
## Fitting models estimated nugget<br>
ml.n ← likfit(s101, ini = c(1,0.5), nug = 0.5)
reml.n ← likfit(s101, ini = c(1,0.5), nug = 0.5,
                  method = "RML")
ols.n ← variofit(bin1, ini = c(1,0.5), nugget=0.5,
                  weights="equal")
wls.n ← variofit(bin1, ini = c(1,0.5), nugget=0.5)
##
par(mfrow = c(1,3))
plot(bin1, main =
        expression(paste(tau^2 == 0, "␣␣(fixed)")))
lines(ml, max.dist = 1)
```


```
lines(reml, lwd = 2, max.dist = 1)
lines(ols, lty = 2, max.dist = 1)
lines(wls, lty = 2, lwd = 2, max.dist = 1)
legend(0.5, 0.4,
       legend = c("ML", "REML", "OLS", "WLS"),
       lty = c(1,1,2,2), lwd = c(1,2,1,2))
##
plot(bin1, main =
    expression(paste(tau^2 == 0.15, "  (fixed)")))
lines(ml.fn, max.dist = 1)
lines(reml.fn, lwd = 2, max.dist = 1)
lines(ols.fn, lty = 2, max.dist = 1)
lines(wls.fn, lty = 2, lwd = 2, max.dist = 1)
legend(0.5, 0.4,
    legend = c("ML", "REML", "OLS", "WLS"),
    lty = c(1,1,2,2), lwd = c(1,2,1,2))
##
```


```
plot(bin1, main =
     expression(paste("estimated  ", tau^2)))
lines(ml.n, max.dist = 1)
lines(reml.n, lwd = 2, max.dist = 1)
lines(ols.n, lty = 2, max.dist = 1)
lines(wls.n, lty =2, lwd = 2, max.dist = 1)
legend(0.5, 0.4,
     legend = c("ML", "REML", "OLS", "WLS"),
     lty = c(1,1,2,2), lwd = c(1,2,1,2))
##
par(par.ori)
##
#save.image()
ml.n
summary(ml.n)
##
## 5. Variogram Envelopes
```


```
##
env.mc <- variog.mc.env(s101, obj.var=bin1)
env.model <- variog.model.env(s101, obj.var=bin1,
                              model=wls)
par(mfrow=c(1,2))
plot(bin1, envelope=env.mc)
plot(bin1, envelope=env.model)
par(par.ori)
##
## 6. Profile likelihood
##
prof <- proflik(ml, geodata = s101,
                sill.val = seq(0.48, 2, l=11),
                range.val = seq(0.1, 0.52, l=11),
                uni.only = FALSE)
par(mfrow=c(1,3))
plot(prof, nlevels=16)
```


```
par(par.ori)
##
## 7. Cross-validation
## zeitaufwendig!!!
##
xv.ml <- xvalid(s101, model=ml)
xv.wls <- xvalid(s101, model=wls)
xvR.ml <- xvalid(s101, model=ml, reest=TRUE)
xvR.wls <- xvalid(s101, model=wls, reest=TRUE,
                  variog.obj=bin1)
#save.image()
##
par(mfcol = c(5,2), mar=c(2.5,2.5,.5,.5),
    mgp=c(1.8,0.8,0))
plot(xv.wls)
par(par.ori)
##
```


```
## 8. Kriging
##
plot(s101$coords, xlim=c(0,1.2), ylim=c(0,1.2))
loci ← matrix(
        c(0.2, 0.6, 0.2, 1.1, 0.2, 0.3, 1.0, 1.1),
        ncol=2)
text(loci, as.character(1:4), cex=1.3, col="red")
polygon(x=c(0,1,1,0), y=c(0,0,1,1), lty=2)
kc4 ← krige.conv(s101, locations = loci, krige =
            krige.control(cov.pars = wls$cov.pars))
## defining the grid
pred.grid ←  expand.grid(seq(0,1, l=51),
                    seq(0,1, l=51))
#pred.grid ←  expand.grid(seq(0,1, l=101),
#                    seq(0,1, l=101))
## kriging calculations
kc ← krige.conv(s101, locations = pred.grid, krige =
```


```
                krige.control(cov.pars = ml$cov.pars))
## displaying predicted values
image(kc, loc = pred.grid, coords = s101$coords,
        col=gray(seq(1,0.1,l=30)))
#save.image()
##
## 9. Bayesian prediction
##
pr ← prior.control(phi.discrete = seq(0, 5, l=101),
        phi.prior="rec")
bsp4 ← krige.bayes(s101, loc = loci, prior = pr,
            output = output.control(n.post=10000))
hist(bsp4$posterior$sample$beta, main="",
        xlab=expression(beta), prob = T)
hist(bsp4$posterior$sample$sigmasq, main="",
        xlab=expression(sigma^2), prob = T)
hist(bsp4$posterior$sample$phi, main="",
```


```
        xlab=expression(phi), prob = T)
plot(bin1, ylim = c(0,2))
lines(bsp4, max.dist = 1.2)
lines(bsp4, max.dist = 1.2, summ = "median",
        lty = 2)
lines(bsp4, max.dist = 1.2, summ = "mean",
        lwd = 2, lty = 2)
legend(0.4, 0.4,
    legend = c("posterior mode", "posterior median",
           "posterior mean"), lty = c(1,2,2),
           lwd = c(1,1,2))
#save.image()
##
par(mfrow=c(2,2), mar=c(3,3,.5,.5), mgp=c(1.5,.7,0))
for(i in 1:4){
  ## curve(dnorm(x, mean=kc4$pred[i],
  ##     sd=sqrt(kc4$krige.var[i])),
```


```
##      from=kc4$pred[i] - 3*sqrt(kc4$krige.var[i]),
##      kc4$pred[i] +3*sqrt(kc4$krige.var[i]))
    kpx <- seq(kc4$pred[i] - 3*sqrt(kc4$krige.var[i]),
               kc4$pred[i] +3*sqrt(kc4$krige.var[i]),
               l=100)
    kpy <- dnorm(kpx, mean=kc4$pred[i],
                 sd=sqrt(kc4$krige.var[i]))
    bp <- density(bsp4$predic$sim[i,])
    rx <- range(c(kpx, bp$x))
    ry <- range(c(kpy, bp$y))
    plot(cbind(rx, ry), type="n",
         xlab=paste("Location", i), ylab="density",
         xlim=c(-4, 4), ylim=c(0,1.1))
    lines(kpx, kpy, lty=2)
    lines(bp)}
  par(par.ori)
## definig grid
```


```
pred.grid ← expand.grid(seq(0,1, l=51),
                        seq(0,1, l=51))
#pred.grid ← expand.grid(seq(0,1, l=101),
#                        seq(0,1, l=101))
## Bayesian prediction
## sehr zeitaufwendig !!!
bsp ← krige.bayes(s101, loc = pred.grid, prior =
        prior.control(phi.discrete = seq(0,5,l=101)),
        output=output.control(n.predictive=2))
#save.image()
par(mfrow=c(2,2), mar=c(3,3,3,0))
image(bsp, loc = pred.grid, main = "predicted",
        col=gray(seq(1,0.1,l=30)))
image(bsp, val ="variance", loc = pred.grid,
        main = "prediction variance",
        col=gray(seq(1,0.1,l=30)))
image(bsp, val = "simulation", number.col = 1,
```


```
    loc = pred.grid, main =
    "a simulation from\nthe predictive distribution",
    col=gray(seq(1,0.1,l=30)))
image(bsp, val = "simulation", number.col = 2,
    loc = pred.grid, main =
    "another simulation from \n
     the predictive distribution",
    col=gray(seq(1,0.1,l=30)))
par(par.ori)
##
## 10. Simulation
##
sim1 ← grf(100, cov.pars=c(1, .25))
points.geodata(sim1, main=
        "simulated locations and values")
plot(sim1, max.dist=1, main=
        "true and empirical variograms")
```


```
sim2 ← grf(441, grid="reg", cov.pars=c(1, .25))
image(sim2, main="a␣\"smallish\"␣simulation",
      col=gray(seq(1, .1, l=30)))
sim3 ← grf(40401, grid="reg", cov.pars=c(10, .2),
           met="circ")
image(sim3, main=
      "a␣much␣finer␣grid␣for␣the␣␣simulation",
      col=gray(seq(1, .1, l=30)))
##
##
##
```


L.17 Lösungsvorschläge Blatt 17

L.17.1 Entwurf einer Simulationsklasse

L.17.2 Indexoperator

L.17.4 Verteilungsklasse II

L.17.5 Simulationsklasse II

L.17.6 Schätzerauswertungsklasse

L.18 Lösungsvorschläge Blatt 18

L.18.1 Checken/Erstellen eines Pakets

L.18.2 Anlegen eines Daten-Pakets

L.18.3 Anlegen eines eigenen R-Pakets

L.18.4 Arbeit mit Sweave

L.19 Lösungsvorschläge Blatt 19

L.19.2 R und MySQL

L.19.3 Aufruf von C Code unter R

L.19.4 Aufruf von Fortran Code unter R

L.19.5 Paralleles Rechnen mit R

Literatur

Literatur

Homepage John Chambers: http://cm.bell-labs.com/cm/ms/who/jmc/pub.html.

R Development Core Team (2006): ***R: A language and environment for statistical computing***. **R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.** http://www.R-project.org.

—— (2006a): ***R: An Introduction to R***. **R Foundation for Statistical Computing, Vienna, Austria.** http://cran.r-project.org/doc/manuals/R-intro.pdf. **8.3.11 (b)**

—— (2006b): ***R: Writing R Extensions***. **R Foundation for Statistical Computing, Vienna, Austria.** http://cran.r-project.org/doc/manuals/R-exts.pdf. **3.4**, **8.2.3**, **8.2.6**, **8.2.6 (j)**, **8.3.2**, **8.3.5**, **8.3-4**, **8.3-6**, **8.3.5 (f)**, **8.3.6**, **8.4.4 (b)**

—— (2006c): ***R: R Language Definition***. **R Foundation for Statistical Computing, Vienna, Austria.** http://cran.r-project.org/doc/manuals/R-lang.pdf. **3.4**, **3.6.3**

—— (2006d): ***R: R Installation and Administration***. **R Foundation for Statistical Computing, Vienna, Austria.** http://cran.r-project.org/doc/manuals/R-admin.pdf. **8.2.10**, **8.2-3**, **8.3.12 (a)**

—— (2006e): **R: Manuals-Homepage. R Foundation for Statistical Computing, Vienna, Austria.** http://cran.r-project.org/manuals.html.

—— (2006f): **R: weitere Dokumentation.** http://cran.r-project.org/other-docs.html.

Homepage Auckland Workshop: (2003): http://www.stat.auckland.ac.nz/S-Workshop/.

Baron, J.(2005): "R ***reference card***". **Zu beziehen unter** http://cran.r-project.org/doc/contrib/refcard.pdf **0.6**

Short, T.(2005): "R ***reference card***". **Zu beziehen unter** http://cran.r-project.org/doc/contrib/Short-refcard.pdf **0.6**

Ruckdeschel, Kohl, Stabla and Camphausen(2006): Homepage zu Paket "distr": http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR.

Aho A.V., Sethi R. and Ullman J.D. (1988): *Compilerbau. Teil 1. (Compiler construction. Pt 1)*. Internationale Computer-Bibliothek. Addison-Wesley Publishing Company, Bonn etc. 1.10-1

Anderson T. (1984): *An introduction to multivariate statistical analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, Inc., 2. Aufl. 7.2.3 (c)

Bates, D. (2003): *Using external routines*. http://www.stat.wisc.edu/courses/st771-bates/slides/wk3-4.pdf. 8.3.5 (d)

Bavington, M. (2003): Debugging Without (Too Many) Tears.. *R-News*, 3(3): 29–32. 3.4

Becker R.A., Chambers J.M. and Wilks A.R. (1988): *The new S language. A programming environment for data analysis and graphics.*. Wadsworth & Brooks/Cole Computer Science Series. Wadsworth & Brooks/Cole Advanced Books & Software., Pacific Grove, CA. 8.1.3 (a)

Belsley D.A., Kuh E. and Welsch R.E. (1980): *Regression diagnostics: identifying influential data and sources of collinearity*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics Section. Chapman and Hall. 7.1.1 (e)

Bengtsson H. (2003): The R.oo package - object-oriented programming with references using standard R code. In: Hornik K., Leisch F. and Zeileis A. (Eds.) *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Vienna, Austria. Published as http://www.ci.tuwien.ac.at/Conferences/DSC-2003/. 8.1.3 (c)

Booch G. (1995): *Objektorientierte Analyse und Design. (Object oriented analysis and design)*. Addison-Wesley., 1., korrigierter Nachdruck Deutsche Übersetzung. 8.1.1 (a)

Brockwell P.J. and Davis R.A. (1991): *Time series: theory and methods*. Springer Series in Statistics. Springer, 2. Aufl. 7.3.1 (b)

——— (2002): *Introduction to time series and forecasting*. Springer Texts in Statistics. Springer, 2. Aufl. 7.3.1 (b)

Chambers J.M. (1993a): Classes and methods in S. I: Recent developments. *Comput. Stat.*, 8(3): 167–184. http://cm.bell-labs.com/stat/doc/93.26.ps. 8.1.3 (a)

——— (1993b): Classes and methods in S. II: Future directions. *Comput. Stat.*, 8(3): 185–196. http://cm.bell-labs.com/stat/doc/93.27.ps. 8.1.3 (a)

——— (1998): *Programming with data. A guide to the S language*. Springer. Siehe auch http://cm.bell-labs.com/stat/Sbook/index.html. 0.1.4, 0.6, 8.1.3 (a), 8.1-2, 8.1-6, 8.1.7, 8.1.9, 8.3.10 (b), 8.3.10 (c)

Chambers J.M. and Hastie T.J. (1992): *Statistical models in* S. Wadsworth & Brooks/Cole Computer Science Series. Wadsworth & Brooks/Cole Advanced Book & Software., Pacific Grove, CA. 8.1.3 (a)

Christensen R. (1996): *Plane answers to complex questions. The theory of linear models*. Springer Texts in Statistics. Springer, 2. Aufl. 7.1-2

Cleveland W.S. (1985): *The Elements of Graphing Data*. Wadsworth. 4.3-2, 4.4.2 (c)

——— (1993): *Visualizing Data*. Hobart Press, Summit, N.J. 4.6

Cliff A. and Ord J. (1981): *Spatial processes: models and applications.*. Pion Ltd., London. 7.4.1 (c)

Cook R. and Weisberg S. (1982): *Residuals and influence in regression*. Monographs on Statistics and Applied Probability. Chapman and Hall. 7.1.1 (e)

Cressie N.A. (1991): *Statistics for spatial data.*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons. 7.4.1 (c)

Dalgaard P. (2002): *Introductory Statistics with* R. Springer. Siehe auch http://www.biostat.ku.dk/ pd/ISwR.html 0.6

Davis J.C. (1986): *Statistics and Data Analysis in Geology*. John Wiley and Sons, 2. Aufl. 7.4.1 (c)

Dolić D. (2004): *Statistik mit R*. Oldenbourg. Siehe auch http://www.dolic.de/R/index.html 0.6

Durbin J. and Koopman S.J. (2001): *Time Series Analysis by State Space Methods*. Oxford University Press. 7.3.1 (b)

Efron B. and Tibshirani R.J. (1993): *An introduction to the bootstrap*, Bd. 57 von *Monographs on Statistics and Applied Probability*. Chapman & Hall. 2.9.3

Fahrmeir L. and Tutz G. (2001): *Multivariate statistical modelling based on generalized linear models*.. Springer Series in Statistics. Springer, 2. Aufl. 7.1.2 (a)

Fahrmeir L., Hamerle A. and Tutz G. (Hrsg.) (1996): *Multivariate statistische Verfahren.(Multivariate statistical methods)*. Springer Series in Statistics. Walter de Gruyter, 2., revidierte Aufl. Unter Mitarbeit von Wolfgang Brachinger, Walter Häußler, Heinz Kaufmann, Peter Kemény, Christian Kredler, Willi Nagl, Friedemann Ost, Heinz Pape.

Faraway J.J. (2002): *Practical Regression and Anova using* R. http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf. 0.6, 7.1-2, 7.1.1 (i)

—— (2004): *Linear Models with* R. CRC Press. Siehe auch http://www.stat.lsa.umich.edu/ faraway/LMR/.

Fletcher R. (1987): *Practical methods of optimization.* Wiley, 2. Aufl. 6.5.4

Flury B. (1997): *A first course in multivariate statistics*. Springer Texts in Statistics. Springer. 7.2.3 (c)

Flury B. and Riedwyl H. (1983): *Angewandte multivariate Statistik. Computergestuetzte Analyse mehrdimensionaler Daten*. Gustav Fischer Verlag, Stuttgart - New York. 7.2.3 (c)

Fox J. (2002): *An* R *and* S-Plus *Companion to Applied Regression*. Sage Publications, Thousand Oaks, CA, USA. http://www.socsci.mcmaster.ca/jfox/Books/Companion/. 0.6, 8.4.3 (b)

Geiger C. and Kanzow C. (1999): *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben. (Numerical methods for solution of unconstrained optimization problems)*. Springer. 6.5.4

—— (2002): *Theorie und Numerik restringierter Optimierungsaufgaben. (Theory and numerics of constrained problems of optimization)*.. Springer. 6.5.4

Geist G.A., Kohl, J.A. and Papadopoulos P.M. (1996): PVM and MPI: A Comparison of Features. *Calculateurs Paralleles*, 8(2) http://www.csm.ornl.gov/pvm/PVMvsMPI.ps 8.3-8

Gentleman R. (2002): *S4 Classes in 15 Pages, more or less*. http://www.biostat.harvard.edu/courses/individual/bio271/lectures/L11/S4Objects.pdf. 0.1.4

—— (2003): *Object Orientated Programming. Slides of a Short Course held in Auckland*. http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/Methods.pdf.

Georgii H.O. (2002): *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik*. de Gruyter Lehrbuch. de Gruyter. 2.3-15

Gouriéroux C. (1997): *ARCH Models and financial applications*.. Springer Series in Statistics. Springer. 7.3.1 (b)

Granger C. and Newbold P. (1986): *Forecasting economic time series*.. Economic Theory, Econometrics, and Mathematical Economics. Academic Press, Inc. (Harcourt Brace Jovanovich, Publishers), 2. Aufl. 7.3.1 (b)

Hamilton J.D. (1994): *Time series analysis*. Princeton University Press. 7.3.1 (b), 7.3.5 (a)

Hammersley J. and Handscomb D. (1964): *Monte Carlo methods*. Methuens Monographs on Applied Probability and Statistics. Methuen & Co. Ltd. / John Wiley & Sons Inc. , London / New York. 2.3.2 (g)

Hampel F.R., Ronchetti E.M., Rousseeuw P.J. and Stahel W.A. (1986): *Robust statistics. The approach based on influence functions*. Wiley Series in Probability and Mathematical Statistics. Wiley. 5.2.3 (e)

Härdle W., Müller M., Sperlich S. and Werwatz A. (1998): *Non– and Semiparametric Modelling*. http://wotan.wiwi.hu-berlin.de/statistik/lehrmaterial/statmat_e.html. 2.7, 2.7.3 (d)

Härdle, W. (1991a): *Applied nonparametric regression*, Bd. 19 von *Econometric Society Monographs*. Cambridge University Press. 2.7

—— (1991b): *Smoothing techniques. With implementation in S*. Springer Series in Statistics. Springer-Verlag. 2.7.2

Härdle, W. and Simar, L. (2003): *Applied Multivariate Statistical Analysis*. Springer. http://www.quantlet.com/mdstat/scripts/mva/pdf/mvapdf.pdf. 7.2.3 (c), A.15.5

Harvey A. (1993): *Time series models*. Harvester Wheatsheaf, 2. Aufl. 7.3.1 (b)

Huber P.J. (1977): *Robust statistical procedures*, Bd. 27 von *CBMS - NSF Regional Conference Series in Applied Mathematics*.. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pennsylvania. 5.2.3 (e)

—— (1981): *Robust statistics*. Wiley Series in Probability and Mathematical Statistics. Wiley. 5.2.3 (e)

Ihaka R. and Gentleman R. (1996): R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3): 299–314.

Isaaks E. and Srivastava R. (1989): *An introduction into applied geostatistics*. Oxford university press. 7.4.1 (c)

Knuth D.E. (1998): *The art of computer programming, Bd. 2 – Seminumerical algorithms*. Addison-Wesley, Bonn, 3. Aufl. 2.3.2 (f), 2.3.2 (g)

Krause A. and Olson M. (2000): *The basics of S and S-Plus*. Statistics and Computing. Springer, 2. Aufl.

Ligges U. (2005): *Programmieren mit* R. Springer Texts in Statistics. Springer. Siehe http://www.statistik.uni-dortmund.de/~ligges/PmitR/ 0.6, 8.2.7

Luenberger D. (1969): *Optimization by vector space methods (Series in Decision and Control)*. John Wiley and Sons, Inc. 6.5.4

Luenberger D.G. (1984): *Linear and nonlinear programming*. Addison-Wesley Publishing Company, 2. Aufl. 6.5.4

Mardia K.V., Kent J.T. and Bibby J.M. (1979): *Multivariate analysis*. Probability and Mathematical Statistics. Academic Press. 7.2.3 (c)

Marsaglia G. (1997): *A random number generator in C.*. Discussion paper; posting on Usenet newsgroup 'sci.stat.math'. 2.3.2 (f)

Marsaglia G. and Zanan A. (1994): Some portable very-long-period random number genrators. *Computers in Physics*, 8: 117–121. 2.3.2 (f)

Matsumoto M. and Nishimura T. (1998): Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1): 3–30. 2.3.2 (f)

McCullagh P. and Nelder J. (1989): *Generalized linear models*, Bd. 37 von *Monographs on Statistics and Applied Probability*. Chapman and Hall, 2. Aufl. 7.1.2 (a)

Na Li M. and Rossini A.J. (2001): RPVM: Cluster Statistical Computing in R. *R News*, 1(3): 4–7. Siehe http://CRAN.R-project.org/doc/Rnews/. 8.3.12 (c), 8.3-8

Nagel M., Wernecke K.D. and Fleischer W. (1994): *Computergestützte Datenanalyse. (Computer supported data analysis)*. Verlag Technik, München. 7.1.1 (h)

Nolan D. and Speed T. (2000): *Stat Labs. Mathematical statistics through applications*. Springer Texts in Statistics. Springer.

Peng, R. (2002): *An Introduction to the Interactive Debugging Tools in R*. http://www.biostat.jhsph.edu/ rpeng/docs/R-debug-tools.pdf. 3.4, 3.4-3

Pinheiro J.C. and Bates D. (2000): *Mixed-Effects Models in* S *and* S-Plus. Springer Texts in Statistics. Springer. http://nlme.stat.wisc.edu/MEMSS/. 8.4.3 (b)

Press W.H., Teukolsky S.A., Vetterling W.T. and Flannery B.P. (1992): *Numerical recipes in C. The art of scientific computing*. Cambridge Univ. Press, 2. Aufl. 6.4.3, 6.4.4, 6.5.4, 6.5.5

Rieder H. (1994): *Robust asymptotic statistics*. Springer Series in Statistics. Springer. 5.2.3 (e), 1

Ripley B.D. (1981): *Spatial statistics*.. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons. 7.4.1 (c)

—— (1987): *Stochastic simulation*. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. John Wiley & Sons. 0.1.4, 2.3-7, 2.3-8, 2.3.2 (g), 2.8.3

—— (1991): *Statistical inference for spatial processes*. Cambridge University Press. 7.4.1 (c)

—— (2004): Lazy Loading and Packages in R 2.0.0. *R News*, 4(2): 2–5. Siehe http://CRAN.R-project.org/doc/Rnews/. 8.2.4

Rossini A.J., Tierney L. and Na Li, M. (2003): Simple Parallel Statistical Computing in R. *UW Biostatistics Working Paper Series. University of Washington* 193. http://www.bepress.com/uwbiostat/paper193. 8.3.12 (c), 8.3-8

Rothman P.e. (1999): *Nonlinear time series analysis of economic and financial data*, Bd. 1 von *Dynamic Modeling and Econometrics in Economics and Finance*. Kluwer Academic Publishers. 7.3.1 (b)

Ruckdeschel P., Kohl M., Stabla T., and Camphausen F. (2006): S4 Classes for Distributions. *R-News*, 6(2): 10–13. http://CRAN.R-project.org/doc/Rnews/. Siehe auch http://www.uni-bayreuth.de/departments/math/org/mathe7/RUCKDESCHEL/pubs/distr.pdf. 0.1.6, 2.2

Sachs L. and Hedderich J. (2006): *Angewandte Statistik. Methodensammlung mit* R. Springer. (Taschenbuch) 12., vollst. neu bearb. Aufl. 0.6

Sawitzki G. (2005): Einführung in S. Zu beziehen unter http://www.statlab.uni-heidelberg.de/projects/s/s.pdf.

Schlittgen R. and Streitberg B.H.J. (1987): *Zeitreihenanalyse. (Time series analysis)*. R. Oldenbourg Verlag, München-Wien, 2. Aufl. 7.3.1 (b)

Shumway R.H. and Stoffer D.S. (2000): *Time series analysis and its applications.*. Springer Texts in Statistics. Springer. 7.3.1 (b)

Silverman B. (1986): *Density estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability. Chapman and Hall. 2.7

Stoer J. (1999): *Numerische Mathematik. 1: Eine Einführung - unter Berücksichtigung von Vorlesungen von F. L. Bauer. (Numerical mathematics. 1: An introduction - under consideration of lectures by F. L. Bauer)*. Springer, 8., revidierte underweiterteAufl. 6.1.2, 6.4.3

Stoyan D., Kendall W.S. and Mecke J. (1995): *Stochastic geometry and its applications.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons Ltd., 2. Aufl. 7.4.1 (c)

Stroustrup B. (1987): *Die C++ Programmiersprache. (The C++ programming language)*. Internationale Computer-Bibliothek. Addison-Wesley Verlag. Deutsche Übersetzung. 8.1.1 (a)

Tierney L. (2003b): Namespace Management for R. *R News*, 3(1): 2–6. Siehe http://CRAN.R-project.org/doc/Rnews/. 8.2.7

—— (2003b): *Notes on the Generational GC for R*. http://www.stat.uiowa.edu/~luke/R/bytecode.html. 8.3.2, 8.3.10 (a)

—— (2003c): *Notes on Compilation in R*. http://www.stat.uiowa.edu/~luke/R/gengcnotes.html.

Unwin D. (1981): *Introductory spatial analysis*. Methuen, New York, London. 7.4.1 (c)

Venables W. and Ripley B. (1999): *Modern Applied Statistics with S-Plus*. Statistics and Computing. Springer, 3. Aufl. inzwischen (2002) ist die 4. Aufl. erhältich; siehe dazu auch http://www.stats.ox.ac.uk/pub/MASS4/ 2, 0.1.3, 0.1.4, 0.2, 0.4, 0.6, 0.7, 165, 1.10.4, 1.10.4, 2.6, 3, 3.4-4, 3.6.1 (c), 3.6-3, 4.4-1, 7.1.1 (c), 7.1.1 (e), 7.2.5, 8.4.3 (b)

—— (2000): *S Programming*. Statistics and Computing. Springer. Siehe dazu auch http://www.stats.ox.ac.uk/pub/MASS3/Sprog/ 0.6, 8.4.3 (b)

von Alemann H. (1984): *Der Forschungsprozess. Eine Einführung in die Praxis der empirischen Sozialforschung*. Studienskripten zur Soziologie. Teubner, Stuttgart, 2 Aufl. (document)

Wei W.W.S. (1990): *Time series analysis. Univariate and multivariate methods*. Addison-Wesley. 7.3.1 (b)

Wichmann B. and Hill I. (1982): Algorithm AS 183: An Efficient and Portable Pseudo-random Number Generator. *Applied Statistics*, 31: 188–90. Remarks: 34, 198 and 35, 89. 2.3.2 (f)

Yu H. (2002): `Rmpi`: Parallel Statistical Computing in R. *R News*, 2(2): 10–14. Siehe http://CRAN.R-project.org/doc/Rnews/. 8.3.12 (c), 8.3-8