Analyse und Visualisierung von Daten mit R

Datenanalyse unter freier Lizenz

Janko Dietzsch

janko.dietzsch@med.uni-tuebingen.de

Department für Augenheilkunde
Universitätsaugenklinik Tübingen

Chemnitzer Linuxtage 2009

Überblick

Bevor wir starten - „Mission and Prime Directive“

Für statistische Software nach J. Chambers [Cha08]:

- Exploration: The Mission
  „. . . our Mission, as users and creators of software for data analysis, is to enable the best and most thorough exploration of data possible.“
- Trustworthy Software: The Prime Directive
  (to analyze and) „. . . to program in such a way that the computations can be understood and trusted.„

Für diesen Vortrag:

- Mission - möglichst breite Exploration von R
- Erste Direktive - nicht um die Details kümmern und einfach nur den Eindruck mit nach Hause nehmen

Übersicht

R-Biografie des Autors

- Erstkontakt – Anfang 2003
- Praktische Erfahrung bei der Auswertung verschiedener Genexpressionsstudien mit R und BioConductor (BioC)
- Teilnahme an „useR!“ 2004, 2006, 2008
- Betreuung der R-basierten, begleitenden Übungen zu folgenden Lehrveranstaltungen:
  - WS 2003/04 „Microarray Bioinformatik“
  - WS 2006/07 „Microarray Bioinformatik“
- aktuell – Arbeit an einem R-Package für die Ophthalmologie - `Rophta`

Übersicht

Was ist überhaupt R?

Was sagt dazu die Webseite `www.r-project.org`

„R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.“

R - vom Seminarraum in die New York Times [Van09]

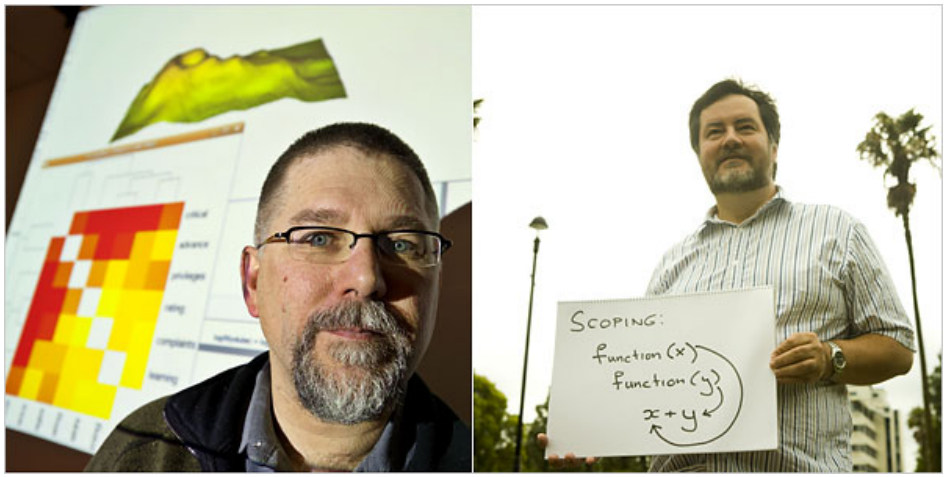


Abbildung: Robert Gentleman (li.) und Ross Ihaka (re.) starten 1992 ihr Projekt und geben 1996 ihr erstes Code-Release frei (entnommen [Van09])

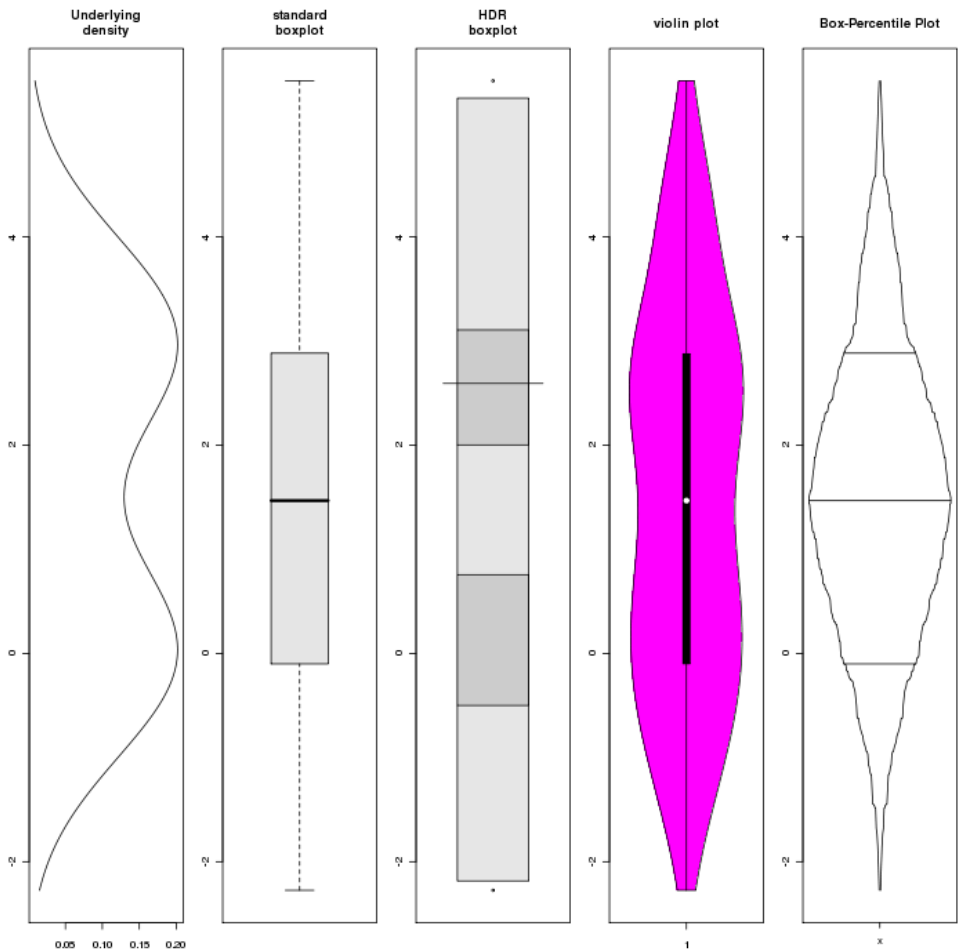
R - vom Seminarraum in die New York Times [Van09]

Zeitachse (teilw. [Lig08] entnommen)

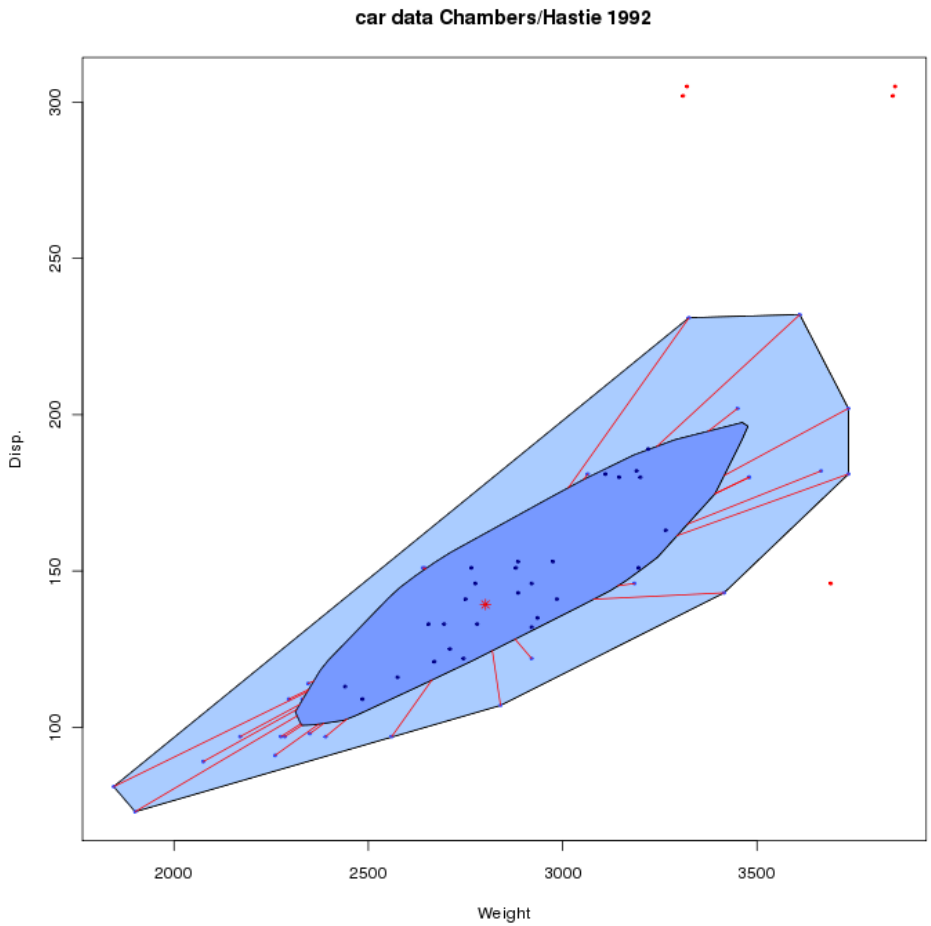
- 1992 Start von R als studentisches Projekt
- 1995 R unter der GPL
- 1997 R Core Team
- 1998 CRAN (Comprehensive R Archive Network)
- 1999 Erste DCS(Distributed Statistical Computing)-Konferenz in Wien
- 2002 R-Foundation
- 2004 Erste R-Konferenz „useR!2004“ in Wien
- 2009 Erwähnung bzw. Artikel in der New York Times[a]

[a] A. Vance, „Data Analysts Captivated by R's Power“, New York Times, 07.01.2009
http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html

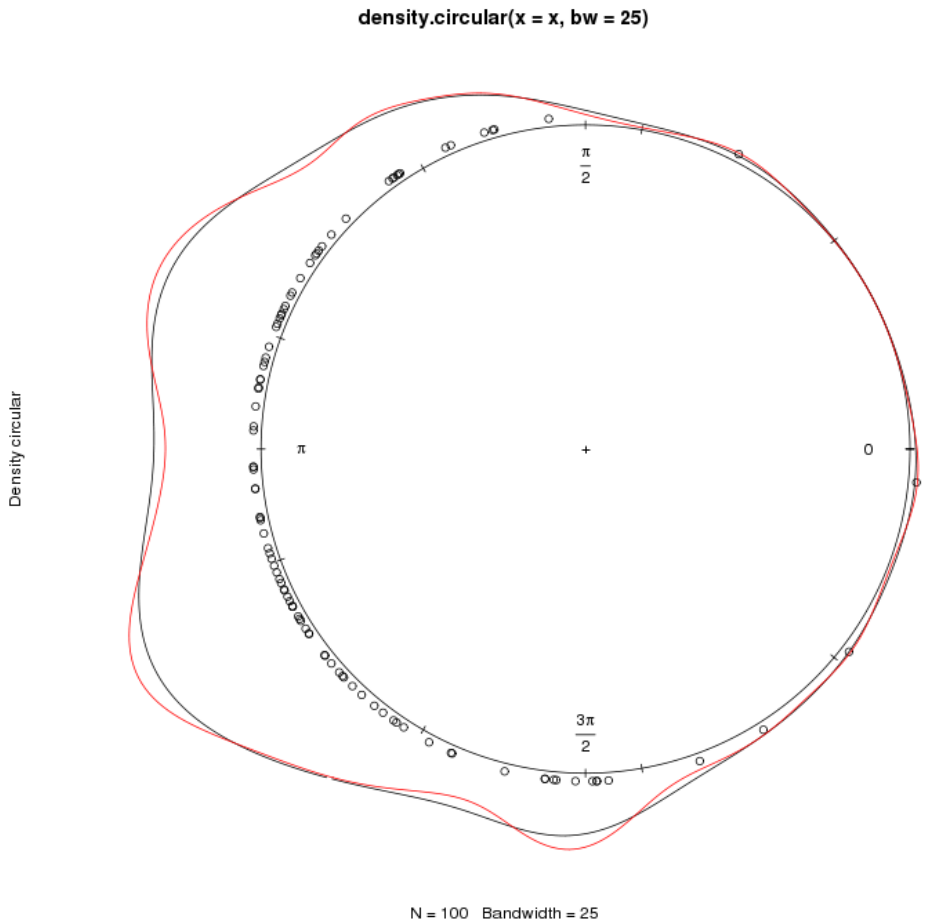
R auf einen Blick . . . Boxplots (Quelle für die folgenden Plots: addictedtor.free.fr/graphiques/)



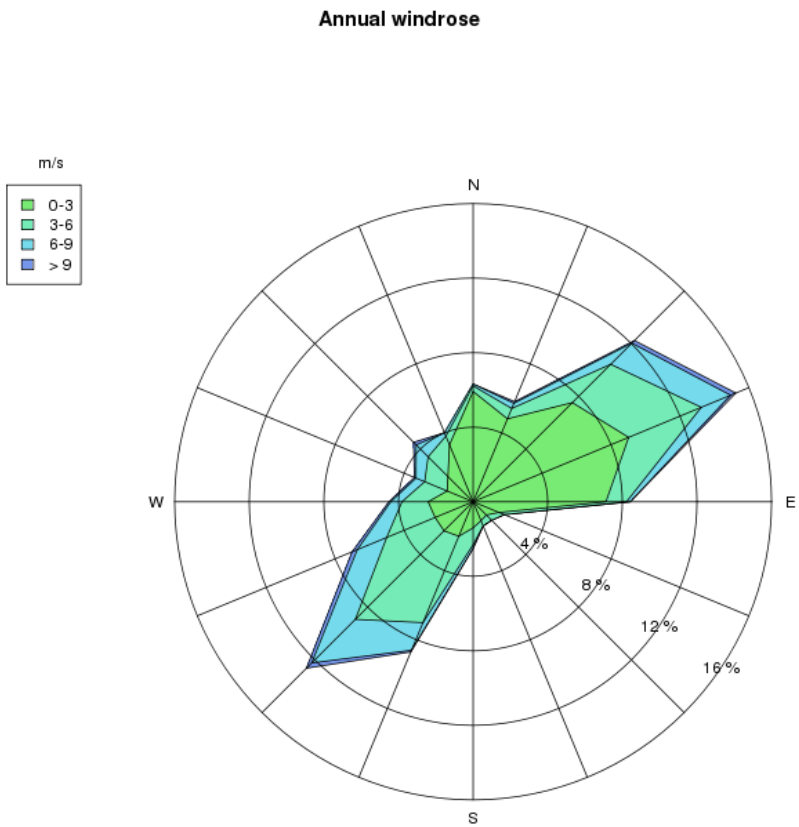
R auf einen Blick . . . Bagplot



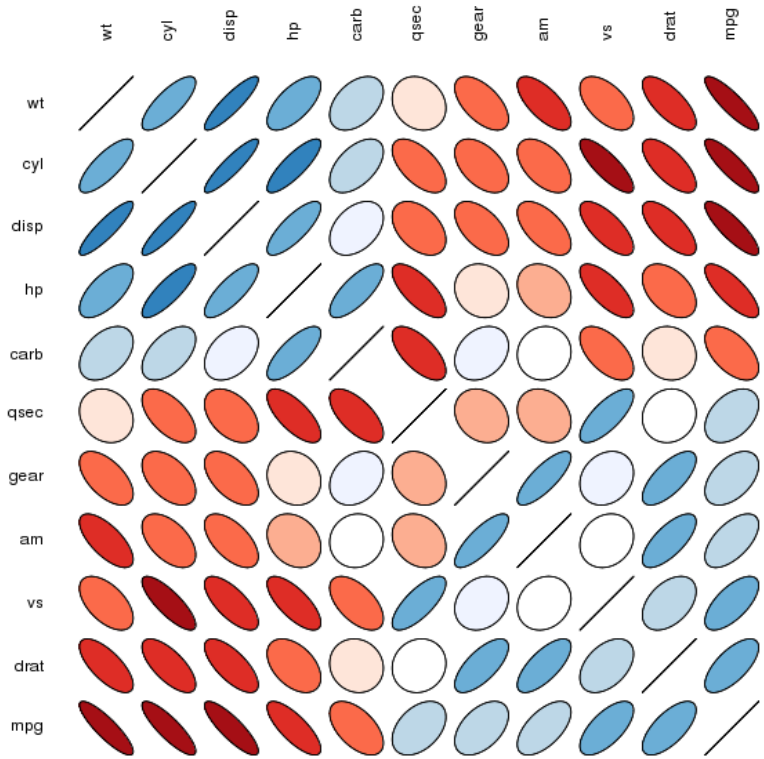
. . . ein „zirkuläres Histogramm“



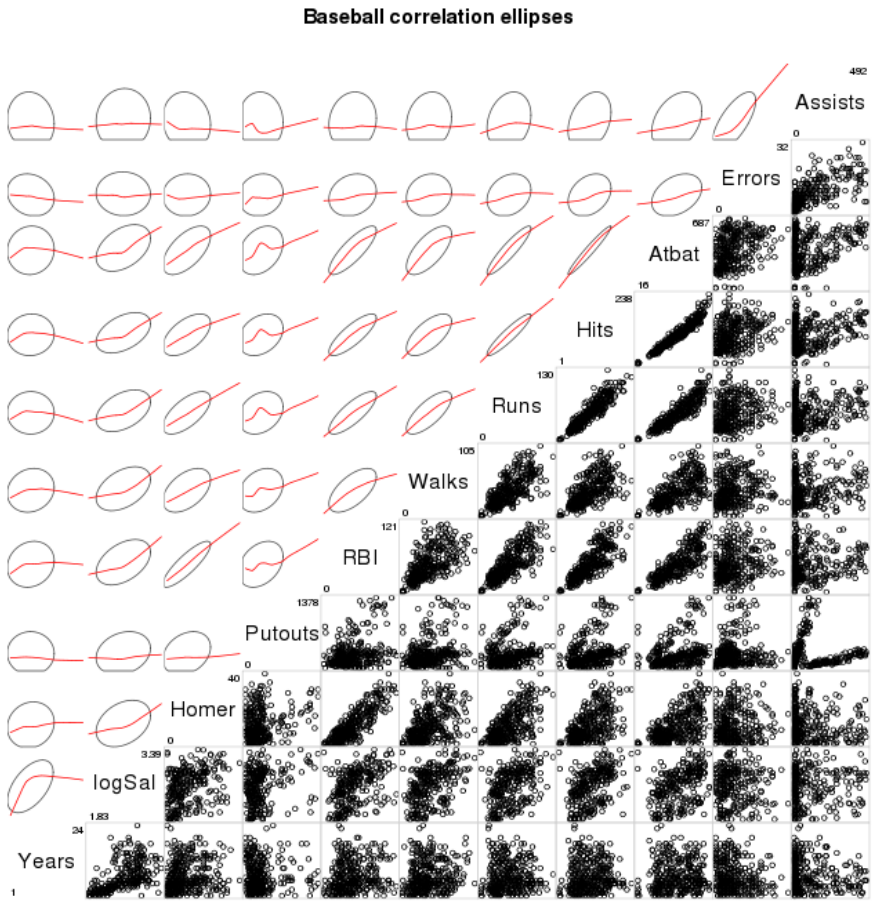
. . . eine Windrose



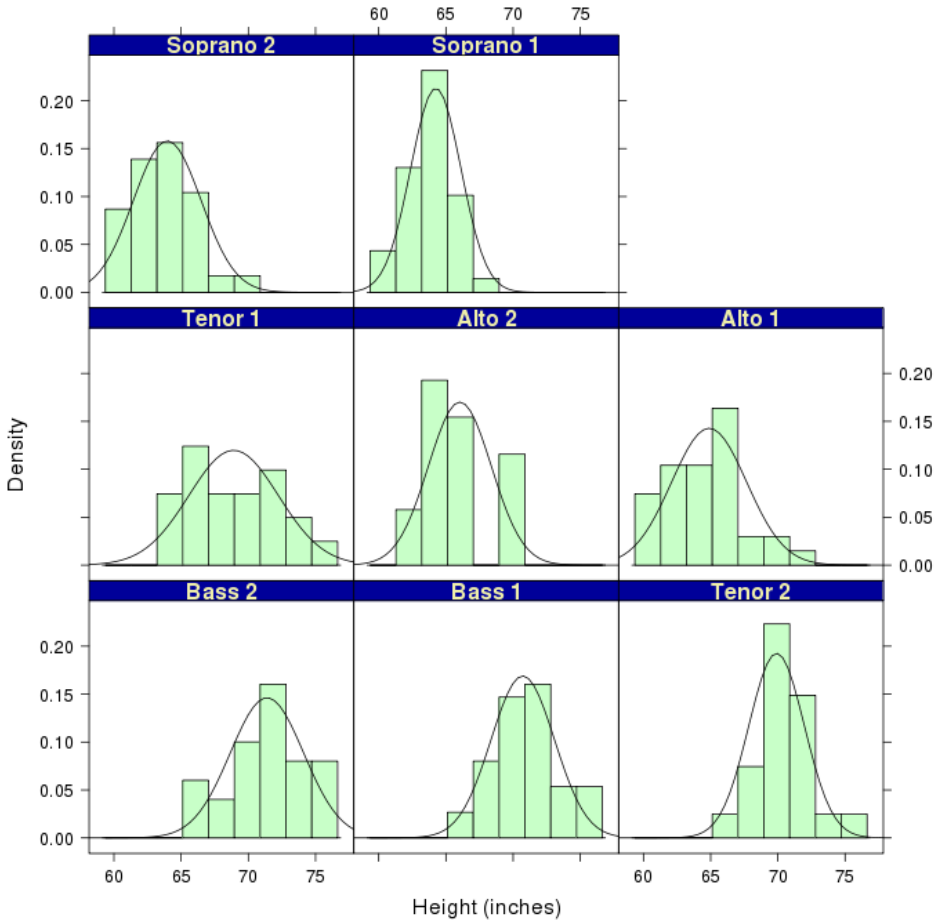
. . . Korrelationsellipsen



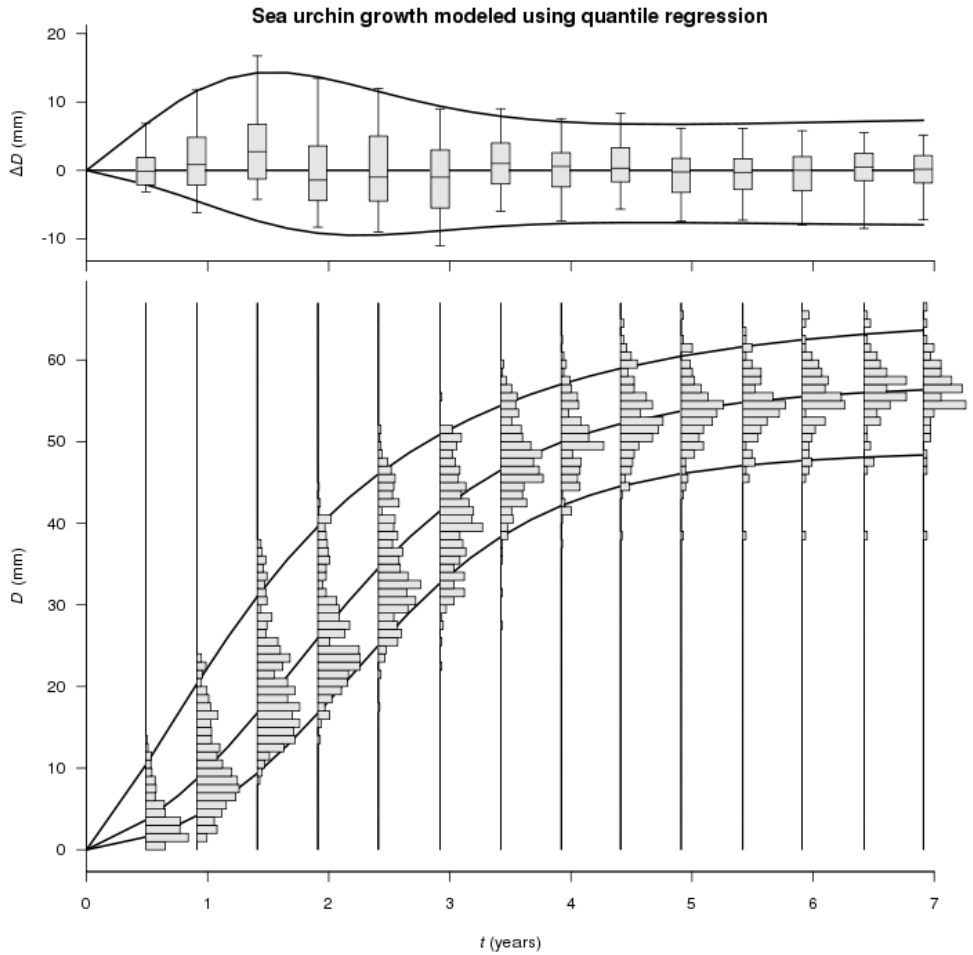
. . . Corrgrams


Baseball correlation ellipses
Assists
Errors
Atbat
Hits
Runs
Walks
RBI
Putouts
Homer
logSal
Years

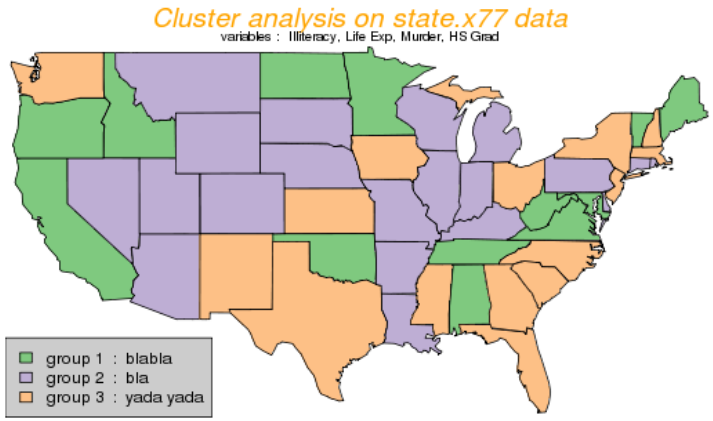
. . . Histogramme konditioniert

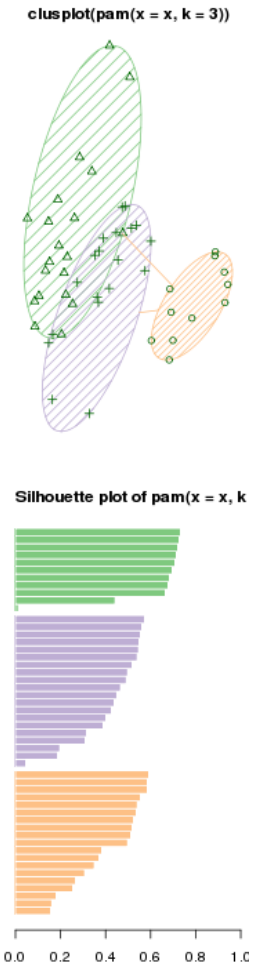


. . . eine Quantil-Regression

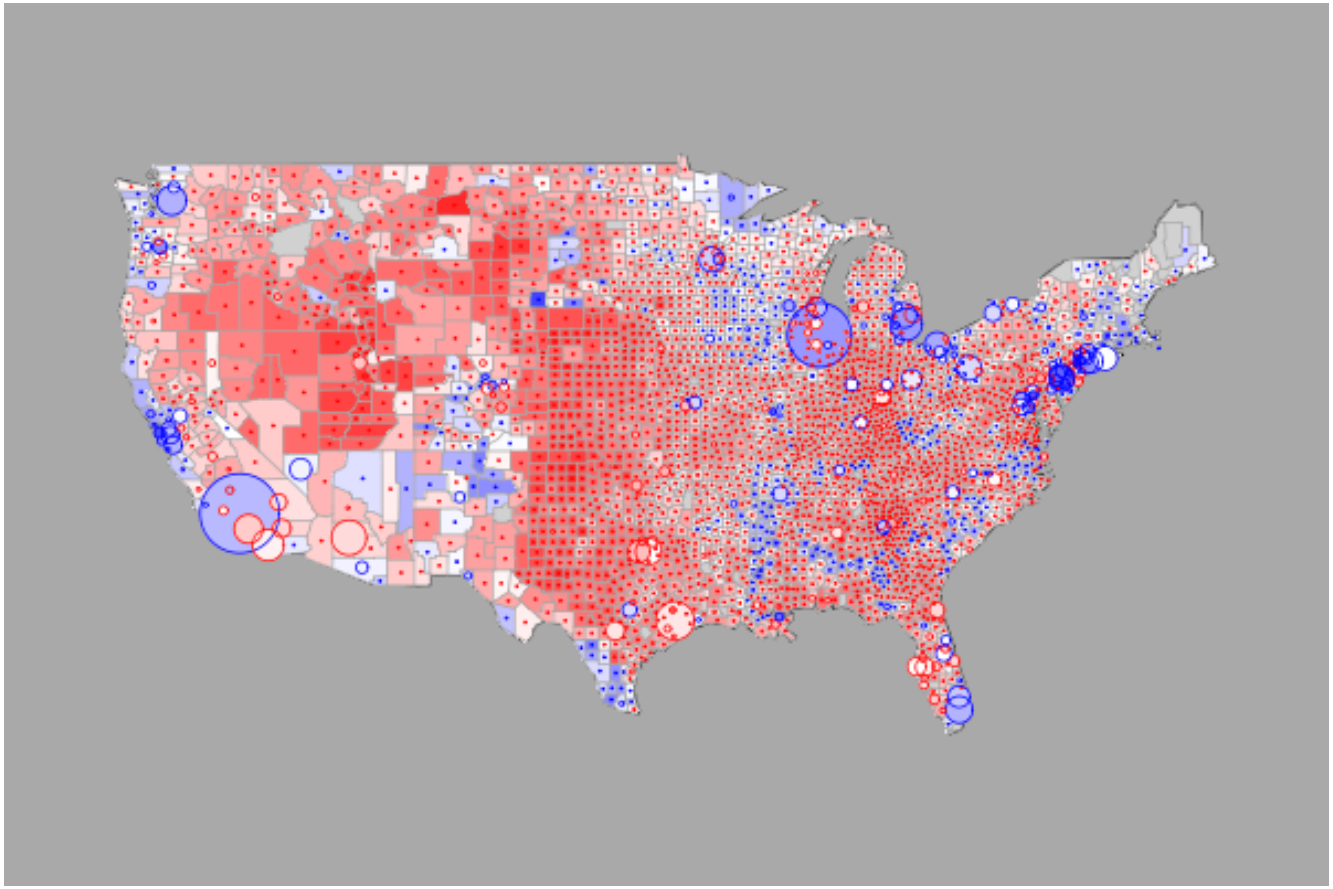


. . . geografische Darstellung statistischer Daten

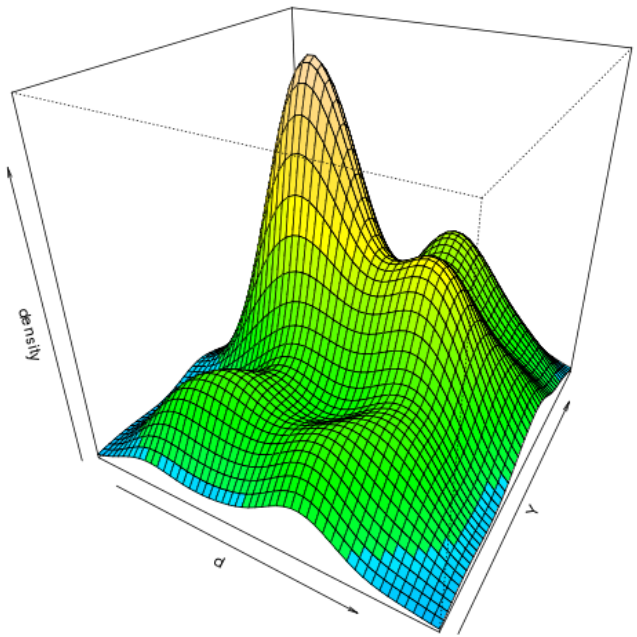


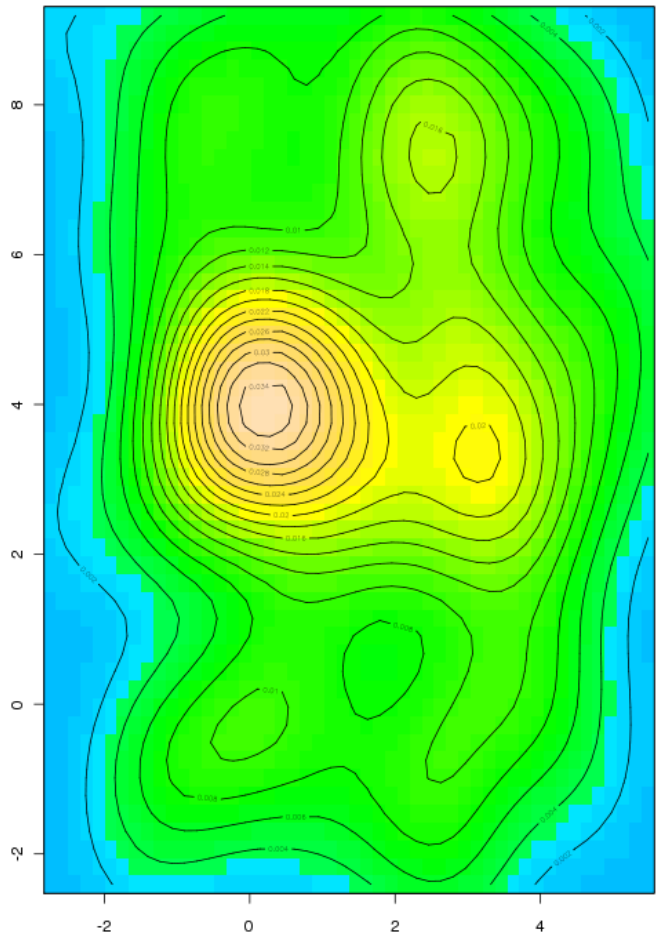


. . . die US-Wahlen von 2004



. . . Kernel Density Estimator





Übersicht

Stärken und Schwächen aus der persönlichen Sicht

- Stärken:
  - Frei
  - Umfassendes Packet/Sprache zur Statistik, Mathematik, Datenvisualisierung ...
  - Große Menge bereits vorhandener Packages
  - Gut erweiterbar durch:
    - eigene R-Skripte
    - native Routinen in C/C++, Fortran, Java ...
    - R-Package-Mechanismus
- Schwächen:
  - Steile Lernkurve
  - Teilweise noch fehlende Funktionalität für bestimmte Fälle
  - Interaktive Grafik ausbaufähig
  - fehlende Zertifizierungen

Literatur- und Online-Quellen I

Projektseite `www.r-project.org`

- Manuals `cran.r-project.org/manuals.html`:
  - An Introduction to R
  - The R language definition
  - Writing R Extensions
  - R Data Import/Export
  - R Installation and Administration
  - R Internals
  - The R Reference Index
- CRAN `cran.r-project.org`
- FAQ `cran.r-project.org/faqs.html`
- Wiki `wiki.r-project.org/rwiki/doku.php`
- RNews `www.r-project.org/doc/Rnews/`

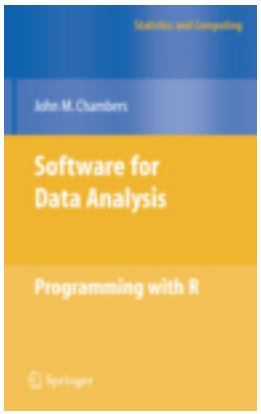
Literatur- und Online-Quellen II

Sonstige Online-Quellen

- Verzeichnis aller Mailinglisten (`R-help`, ...) unter `www.r-project.org/mail.html`
- Paul Murrel's einführendes Buch „**Introduction to Data Technologies**" zu HTML, XML, Datenbanken, SQL, regulären Ausdrücken, und R unter `www.stat.auckland.ac.nz/~paul/ItDT/`
- R.A. Muenchen „**R for SAS and SPSS Users**" `rforsasandspssusers.googlepages.com/`

Spezielle Empfehlung für R-Einsteiger und R-Fortgeschrittene

U. Ligges, **Programmieren mit R**, *Reihe: Statistik und ihre Anwendungen*, Springer Berlin, 2008, 3. überarb. u. erweiterte Aufl., 251 S. 19, ISBN: 978-3-540-79997-9



J. M. Chambers, **Software for Data Analysis: Programming with R**, *Series: Statistics and Computing* Springer Berlin, 2008, 498 Seiten, ISBN: 978-0-387-75935-7

. . . und mehr von der statistischen Seite

Springer Serie zu R „Use R!“– umfaßt mittlerweile ca. 11 verschiedene Titel

L. Sachs, J. Hedderich, **Angewandte Statistik**, *Methodensammlung mit R* Springer Berlin, 2009,13., aktualisierte u. erw. Aufl., ISBN: 978-3-540-88901-4

Grundlegende Konzepte in R (J. Chambers) [Cha08]

- Funktionale Programmierung
- OOP
- Data Frames
- Open Source
- Komponenten bzw. Modularisierung - Core/Packages
- Algorithmen und Interfaces

Einfacher Erstkontakt - eine Beispielsitzung

Arbeitsumgebung

- Konsole
- Workspace
- History
- Image

Einfacher Erstkontakt - Hilfe!

Hilfesystem(e)

- `apropos("so")` – Suche nach Befehlen, die ein „so“enthalten
- `?solve` – Was macht der Befehl „solve“?
- `help.start()` – Öffnet ein Browserfenster mit den Tutorials und dem Hilfesystem
- Durchsuchen der Referenz nach statistischen Stichworten
- Google
- `demo()` – Demos zu einzelnen Packeten ansehen
- `vignette()` – Vignetten, Tutorials als PDF-Dokumente mit R-Code und zugehörigen Erklärungen (Sweave)
- Jemand fragen, der sich damit auskennt. ;o)

Einfacher Erstkontakt - bessere Umgebungen/Editoren

- **E**macs **S**peaks **S**tatistics (ESS) – `ess.r-project.org/`
- **J**ava **G**UI for **R** (JGR) – `jgr.markushelbig.org/JGR.html`, RoSuDa
- Eclipse-Plugin (StatET) – `www.walware.de/goto/statet`

Übersicht

Atomare Datentypen

`mode(x)`

- NULL (`x <- NULL`)
- logical (`x <- TRUE`,`x <- logical(length=0)`)
- numeric (`x <- 3`, `x <- numeric(length=0)`)
- complex (`x <- 3i`, `x <- complex(length=0)`)
- character (`x <- "abcd"`, `x <- character(length=0)`)

Speichermodus – `typeof(x)`

- integer (`x <- integer(3)`)
- double (`x <- 3.2`)

Der zusammengesetzte Datentyp `factor`

`factor()` (und ein Beispiel für die Anwendung von `str()`)

- Prädestiniert zur Darstellung kategorialer Daten
- Anlegen:

```
f <- factor(rep(c("red","blue"),c(2,3)))
```

-

```
> f
[1] red red blue blue blue
Levels: blue red
```

- Stufen (`levels`) intern numerisch kodiert, extern mit den Namen bezeichnet:

```
> str(f)
Factor w/ 2 levels "blue","red": 2 2 1 1 1
```

Datentypen abfagen und konvertieren

Abfrage – `is.XXXX()`

- `a <- 3`
- `is.numeric(a)` → `TRUE`
- `is.character(a)` → `FALSE`
- `is.logical(a)` → `FALSE`

Umwandlung – `as.XXXX()`

- `as.numeric(a)` → `3`
- `as.character(a)` → `"3"`
- `as.logical(a)` → `TRUE`
- `as.logical(0)` → `FALSE`

Datenstrukturen - Vektoren

c(), length(), rep(), seq(), „:“, names()

- `a <- c(34,13,4.5); b <- c("ab","cd"); c <- c(T,T,F)`
- `length(a)` → 3
- `rep(c(2,3,4), c(1,2,3))` → `[1] 2 3 3 4 4 4`
- `seq(from = -5, to = 5, by = 2)` →
  `[1] -5 -3 -1 1 3 5`
- `2:-3` → `[1] 2 1 0 -1 -2 -3`
- `names(a) <- c("Val1","Val2","Val3")` →
  `Val1 Val2 Val3`
  `34.0 13.0 4.5`

Datenstrukturen - Vektoren

Indexzugriff – [], %in%-Operator, as.vector(), is.vector()

- `str(a)`
  ```
  Named num [1:3] 34 13 4.5
  - attr(*, "names")= chr [1:3] "Val1" "Val2"
  "Val3"
  ```
- `a[c(2,3)]` ≃ `a[c(F,T,T)]` ≃ `a[a < 20]` ≃ `a[c("Val2","Val3")]`
- `c("ab","ca","ab","aa","ca","aa") %in% c("aa","ab")`
  ```
  [1] TRUE FALSE TRUE TRUE FALSE TRUE
  ```
- `as.vector(), is.vector()`

Datenstrukturen - Matrizen

matrix(),nrow(),ncol(), dim(), [], as.matrix(), is.matrix(), ...

- `a <- matrix(1:15, nrow = 3, ncol = 5, byrow = FALSE)`
- `nrow(a)` → 3, `ncol(a)` → 5, `dim(a)` → `c(3,5)`
- `a[1,]` → erste Zeile der Matrix
- `a[,2]` → zweite Spalte der Matrix
- Indexzugriff ähnlich wie bei Vektoren, auch „Indexmatrizen“ sind möglich
- Funktionen wie `diag()`, `col()`, `row()`, `solve()`, `eigen()`, `svd()`, `%*%` ...

Datenstrukturen - Array, Listen

array(), as.array(), is.array(), ...

```
a <- array(1:24, dim = c(2,3,4))
```

list(), [], [[]], $, as.list(), is.list(), ...

- Sehr flexibel – beliebige Datenstrukturen, beliebigen Typs können enthalten sein
- Vielen Objekten in R liegt eine Liste zugrunde
- `a <- list(a=c("aa","ab"), b=1:7)`
- Extraktion mit Namen: `a["a"]` $\simeq$ `a[1]` $\simeq$ `a[c(T,F)]`
- Extraktion ohne Namen: `a[["a"]]` $\simeq$ `a[[1]]` $\simeq$ `a$a`

Datenstrukturen - Data Frames

data.frame(), nrow(), ncol(), dim(), [], as.matrix(), is.matrix(), ...

- „Matrixförmige“ Liste
- `a <- data.frame(id = 1:4, gender = c("f","m","m","f"), value = c(5.5,2.5,2.7,5.2))`
- `nrow(a)` → 4, `ncol(a)` → 3, `dim(a)` → `c(4,3)`
- `rownames(a) <- c("a","b","c","d")`
- `a[1,]` ≃ `a["a",]` → erste Zeile
- `a[,2]` ≃ `a["gender"]` ≃ `a$gender` → zweite Spalte
- Indexzugriff ähnlich wie bei Matrizen
  `a[a$gender == "m",]`
- Lokale Kopie im WS – `attach(), detach(), ...`

Übersicht

Einlesen von Datenfiles (CSV)

Lesen von CSV-Files – `read.table()`

- Liest Datentabelle in einen Dataframe
- `read.table(file, header = FALSE, sep = , quote = , dec = ”.”, row.names, col.names, na.strings = ”NA”, colClasses = NA)`

Schreiben von CSV-Files – `write.table()`

- `write.table(x, file = , append = FALSE, quote = TRUE, sep = , eol = , na = ”NA”, dec = ”.”,row.names = TRUE, col.names = TRUE`

Abspeichern und Einlesen von R-Objekten – Images

Binär

- Den ganzen Workspace speichern → `save.image(file = ".RData")`
- Objekt `obj` speichern → `save(obj, file = )`
- Objekt oder Workspace laden → `load(file =)`

ASCII

- Sichern der in `list` aufgeführten Objekte → `dump(list, file = )`
- Einlesen der ASCII-Repräsentation in den Workspace → `source(file = )`

Zugriff auf Datenbanken

- Package DBI → Packages RPostgreSQL, RMySQL, ROracle, RSQLite
- Package RODBC
- Übliches Vorgehen:
  - Verbindung öffnen
  - SQL-Query absetzen (`sqlQuery(verb, "select * from tab where x == 'name' ")`)
  - Verbindung schließen

Übersicht

Die üblichen Konstrukte

- Block `{ }`
- `if ( ) { } else { }`
- `repeat { }`
- `while( ) { }`
- `for (i in V) { }`
- `next` und `break`
- Funktionen (und Dreipunktargument):

```
fun <- function(arg = default, ...) {
return()}
```

Funktionen für die Iteration - (*)apply-Familie

Performante, wiederholte Anwendung einer Funktion `fun` auf die Elemente eines Objektes und Zusammenfassung der Ergebnisse

Dataframes, Listen und Vektoren

- `lapply(obj, fun)`
- `sapply(obj, fun)` → möglichst einfachstes Objekt

Matrizen und Arrays

- `apply(obj, MARGIN = 1, FUN = fun)` → Zeilen
- `apply(obj, MARGIN = 2, FUN = fun)` → Spalten
- `apply(array(1:24,c(2,3,4)), MARGIN = c(1,2), FUN = fun)` → Iteration über die letzte Dimension hinweg

OOP - S3 Klassen I

„Listen- und Attribute-orientiertes“ Modell

Methoden

- `class()` – Abfrage und Setzen des Klassenattributes
- `attributes()` – Abfrage und Setzen aller Klassenattribute
- `methods()` – Abfrage aller verfügbaren Methoden zu einer generischen Funktion

Generische Methoden

- `NameGenerischeFunktion.Klassenname`
- Beispiele generischer Funktionen: `print, summary, plot, ...`
- Beispiele konkreter Funktionen: `print.lm , print.glm, summary.lm, plot.lm, ...`

OOP - S3 Klassen II

Vererbung

- `UseMethod()` – Verlinkung an die entsprechende Methode innerhalb der generischen Funktion
- `inherits(obj, "classname")` – Abfrage der „Vererbungslinie“
- `class(obj)` → `[1] "class.spez" "class.allg"`
- `NextMethod()` – Weiterleitung an die Methode der übergeordneten Klasse:

```
summary.class.spez <- function(obj)
NextMethod("summary")
```

OOP++ - S4 Klassen

Wesentlich formaler und strenger und damit näher an den von anderen Sprachimplementierungen gewohnten Konzepten

- `SetClass(), GetClass()` – Definition/Abfrage der Klasse
- `prototype()` – Prototyp einer Klasse
- `new()` – Anlegen eines Objektes einer Klasse (entsprechend dem Protypen)
- `slot(),@` – Slots
- `slotNames()` – Namen der Slots eines Objektes abfragen
- . . . mehr eventuell in der Demonstration

Diverse Hilfsmittel

- `debug(fun); undebug(fun)` - Debuggen von Fuktionen (`Q` zum Verlassen des Deb-Browsers)
- `gc()` – expliziter Aufruf der Garbage Collection
- `Rprof(filename = "Rprof.out", append = FALSE, interval = 0.02, memory.profiling=FALSE)` – Profiling in R
- ...

Übersicht

Modularität durch Packages

- R kommt bereits mit einer Reihe von Packages, wie `base`, `utils`, `stats`, `...`
- Zusätzliche Fuktionalität wird über Packages zur Verfügung gestellt (Komponentenmodell)

Nachinstallieren

- Quelle ein CRAN-Mirror oder sonstiges
- `install.packages()` – direkt von der R-Session aus herunterladen und installieren, Schreibrechte beachten!
- Package im Quellcode herunterladen und mit `R CMD INSTALL pkg` installieren
- `library(pkg)` – in eine R-Session laden

Lineare Modelle als Beispiel

Polynomielle Regression

- Multiple lineare Regression: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Mögliche Umsetzung in R:
  `fit.lm <- lm(y ~ x + I(x^ 2))`
- Intercept ist beim Anpassen des Modells standardmäßig enthalten
- `summary(fit.lm)` $\longrightarrow$ Teststatistik für das Modell
- `plot(fit.lm)` $\longrightarrow$ führt auf 4 diagnostische Plots
- `predict(fit.lm)` $\longrightarrow$ Schätzungen des Modells, sowie Konfidenzintervalle CI, PI ...

Übersicht

Grafikdevices

- Verschiedene Devices:
  - Rastergrafik – `png()`, `jpeg()`, `bmp()`, `tiff()`, ...
  - Vektorgrafik – `postscript()`, `pdf()`, `svg()`, ...
  - OpenGL – `rgl.*()` im Package `rgl`
- Device öffnen (Raster- und Vektorformate)
- Grafik erzeugen
- Device schließen mit `dev.off()`

Grafik mit R

Mehr in der Demonstration

- Traditionell
- Grid-Engine
- Package `lattice`
- Package `rgl`

Versuch eines Resümees

- Der Ist-Zustand:
  R ist das Open-Source-System für die statistische Analyse und Visualisierung von Daten.
- Meine persönliche Wunschvorstellung (die hoffentlich ein Ausblick ist ;o)):
  R wird die integrierende Plattform für statistische, mathematisch und numerische Probleme und ihre Visualisierung.

Zitierte Quellen I

CHAMBERS, JOHN M.: *Software for Data Analysis: Programming with R*.
Statistics and Computing. Springer Berlin, 1. , 2008.

LIGGES, UWE: *Programmieren mit R*.
Springer Berlin, 3. , 2008.

VANCE, ASHLEE: *Data Analysts Captivated by R's Power*.
New York Times, January 2009.

Vielen Dank für Ihre Geduld und Aufmerksamkeit!

. . . und eine sommerliche Impression aus Tübingen

Blick von der Neckarbrücke hinüber zum Hölderlinturm in dem der kranke Friedrich Hölderlin von 1807 bis zu seinem Tod 1843 lebte.