Datenanalyse I + II (WT + FT 2018)

Dr. D. Steuer
steuer@hsu-hh.de, Tel. 2819, H1 R 1397

Rechnergestützte Statistik
Helmut-Schmidt-Universität Hamburg
Fakultät WiSo

Januar 2018

Struktur der Veranstaltung

- Veranstaltung prinzipiell am Rechner!
- Es ist erforderlich, sich mit dem Rechner und der Programmiersprache `R` , dem GUI `RStudio` und dem Paket `knitr` auseinanderzusetzen!
- Sprechstunde ist im Prinzip jederzeit, für ausführliche Beratung ($>$ 10min) bitte telefonisch oder per mail Termin ausmachen.
- Das Skript soll nach Möglichkeit jeweils am Dienstag vor der Veranstaltung im Netz stehen.
- Tel 2819, steuer@hsu-hh.de

Ziel der Veranstaltung

- Umsetzen der Inhalte von Statistik I und II am Rechner.
- Vermitteln einiger weiterer statistischer Analysemethoden in der Theorie.
- Vermitteln der Nutzung eines Werkzeugs (nämlich R) zur zeitgemäßen Anwendung dieser Methoden.
- Vermitteln eines Eindrucks der Methoden und Probleme der praktischen Datenanalyse.
- Vermitteln von Zuversicht, mit echten Daten umgehen zu können!

Methodische Inhalte (I + II)

- *Sehr* kurze Einführung (*crash course*) in das Programm `R`, sowie `RStudio` und `knitr`
- Datenvorbereitung
- Auffrischung und Vertiefung Regression (diagnostische Plots, multiple Regression, p-Wert, adjustiertes R^2)
- Varianzanalyse (ANOVA)
- Clusterverfahren (Diskriminanzanalyse, etc.)
- Entdecken latenter Variablen (Faktoranalyse, Hauptkomponenten)
- Computerintensive Methoden (Simulation, Bootstrap)
- Übergreifend: Reproduzierbarkeit und Präsentation der Ergebnisse!

Herangehensweise

- Mathematisch saubere Einführung von Verfahren, aber auch etliche Beispiele und grafische Verfahren
- Alle Verfahren werden auch im Rechner umgesetzt (R, r-project.org)
- Die Veranstaltung findet möglichst im EDV-Labor statt
- Besonderes Augenmerk auf der Interpretation der Ergebnisse der Verfahren, nicht auf der einfachen (blinden) Anwendung
- Folien jeweils vorlesungsbegleitend als Skript
- Klausur wird auf jeden Fall gut vorbereitet, Probeklausur unter realistischen Bedingungen zu Beginn des FT

Literatur

- Hain, Statistik mit R, RRZN Hannover 2011 (über die Uni erhältlich)
- Dalgaard, Introductory statistics with R, Springer (elektronisch über die Bibliothek verfügbar)
- Faraway, Linear Models in R, Chapmann and Hall
- Ligges, Programmieren in R, Springer (elektronisch über die Bibliothek verfügbar)
- Literatur für den ersten Teil der Vorlesung, Beispiele sind dort zum Teil entnommen
- Yihui Xie (2012). knitr: A general-purpose package for dynamic report generation in R. R package version 0.6. http://CRAN.R-project.org/package=knitr
- Reichhaltige Informationen im Netz!

Was ist Datenanalyse?

- Datenanalyse ist ein *Prozess*, der über die mathematischen Verfahren hinausgeht!
- Schritte in diesem Prozess sind:
  1. Vertraut machen mit den Daten, d.h. Erläuterungen des Datenlieferanten verstehen. Woher kommen die Daten? Sind sie automatisch erfasst (gemessen) oder von Hand erfasst (Umfragen)?
  2. Daten reinigen, d.h. Ausreißer identifizieren, *missing values* eindeutig und einheitlich kodieren.
  3. Die eigentliche Analyse zerfällt in zwei Teile:
     - Die explorative Analyse (Histogramm, Boxplot etc.), Deskription,
     - und die Modellierung (Regression!) und Tests, schließende Statistik.
  4. Präsentation der Ergebnisse, d.h. sinnvolle Auswahl aus den Ergebnissen treffen und stringent und punktgenau aufbereiten.

Aufgabe 1

Installieren Sie `R` und `RStudio` auf Ihrem Rechner oder machen Sie sich im EDV Labor mit dem Programm vertraut. Vollziehen Sie Beispiele der Vorlesung nach!
Alles weitere, z.B. Einlesen von Dateien, wenn es in der Vorlesung nötig wird.

Die Programmiersprache R

- R ist eine (Interpreter-)Sprache und eine Arbeitsumgebung für statistische Grafik und Analyse.
- R liefert in der Standardinstallation bereits eine große Zahl von statistischen und grafischen Verfahren der Datenanalyse und ist darüber hinaus entworfen, um leicht erweiterbar zu sein. Es gibt (letzten Donnerstag) exakt 12000 Erweiterungspakete für alle Aspekte der Datananalyse. (Jan 15 ca. 6000, Jan 16 ca. 7900, Jan 17 ca. 9800)
- Evtl. die größte Stärke von R liegt in der leichten Anfertigung von veröffentlichungsfähigen Plots, inklusive mathematischer Annotationen.
- Das R-Core Team nennt R eine Umgebung für statistische Berechnungen und Grafik.

Die Programmiersprache `R`

- In dieser Vorlesung: (Weitgehende) Beschränkung auf die bereits implementierten Teile.
- `R` ist dann eine Art statistischer (Hochleistungs-)Taschenrechner.
- Mit `RStudio` und `knitr` (und LaTeX) erhalten Sie zusätzlich eine Softwarekomplettausstattung für die Erstellung wissenschaftlicher Arbeiten.

Warum R?

- R ist *Freie Software* (kostenlos und open source).
- R ist Industriestandard!
- R ist plattformunabhängig, d.h. Sie nutzen weiter den Rechner und das Betriebssystem, das sie gewohnt sind, sei es Windows, MacOs oder Unix. Dasselbe gilt für `RStudio`.
- Kein Vendor-Lock-In! Im Gegensatz zu z.B. Stata, MS Office!
- Hervorragende Fähigkeiten: Immer mehr Firmen nutzen R, also bekommen Sie ein Werkzeug an die Hand, das Sie fast sicher im beruflichen Umfeld wieder sehen werden. R hat sich im universitären Bereich zur Standardsoftware entwickelt. Im industriellen Bereich ebenfalls überragende Bedeutung. (Survey of Datamining Tools 2013)
- Am 7.1.2009 der Durchbruch: R auf der Titelseite der NYT!
- *lingua franca* der Statistik!

Warum R?

- Hervorragende eingebaute Hilfefunktion!
- Lokalisiert in etlichen Sprachen.
- Professioneller (oder besser) Support über Mailinglisten!
- Professionelle (oder besser) Qualitätskontrolle der Software ('make check'). Validierung der Software und der Rechenergebnisse während der ganzen Entwicklung.
- Sehr gute Handbücher werden mitinstalliert (Reference Manual $>$ 3000 Seiten).
- Für Bachelor-, Master- oder Doktorarbeiten: sehr gute Integration mit LaTeXund OpenOffice. (MS Office ist auch ok.)
- Hervorragend geeignet für *Reproducible Research*.

Benutzerinterfaces für R

- Im Kern ist R ein Interpreter mit *read-eval-loop*, der über die Kommandzeile bedient wird!
- Empfehlenswert: Interface zu einem externen Editor (emacs (!), winedt, etc.) oder GUI.
- Hier nutzen wir RStudio. Es gibt andere GUIs, diese werden in der Vorlesung nicht behandelt. (Tinn-R, Revolution, Jaguar, rkward)
- Batch mode (skriptgesteuert).
- etwas ausgefallener: R als Modul des Webservers oder als shared library aus anderen Programmiersprachen aufrufen (python, perl).

Einrichtung der Arbeitsumgebung

- Einloggen
- Filebox einhängen
- RStudio starten
- Neues Projekt anlegen: Ordner `DA2018` in Filebox anlegen!

Eine erste R-Sitzung

```
steuer@gaia> R

R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-suse-linux-gnu (64-bit)

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

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

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

Zuerst:

```
> contributors()
### Liste der Entwickler
> citation()
To cite R in publications use:
  R Core Team (2013). R: A language and environment for statistical
  computing. R Foundation for Statistical Computing, Vienna, Austria.
  URL http://www.R-project.org/.
A BibTeX entry for LaTeX users is
  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = {2013},
    url = {http://www.R-project.org/},
  }
We have invested a lot of time and effort in creating R, please cite it
when using it for data analysis. See also 'citation("pkgname")' for
citing R packages.
### Zitierung von R als Literaturstelle. R hat eine ISBN!
```

Erste Schritte: interaktive Nutzung von R

R als Taschenrechner

```
> 3 + 4
[1] 7
> log(0)
[1] -Inf
> log(-1)
[1] NaN
Warning message:
NaNs were generated in: log(x)
> pi  #es kommt auf Groß- oder Kleinschreibunrg an
[1] 3.141593
> wert <- 3
> wert <- wert^2
> x <- .Last.value
###  '=' statt '<-' geht "neuerdings" auch, auch '->'
> ls()
[1] "wert" "x
> rm(x)
> q()
```

Externe Pakete

- In erheblichem Umfang zusätzliche Funktionalität in externen *packages* (oder *views*)
  `available.packages()` gibt eine Liste der aktuell vorhandenen Pakete
- Einfaches Einfügen in eine bestehende R Installation

  ```
  install.packages("faraway")
  install.packages("knitr")
  ```
- Laden in eine laufende R-Sitzung mit `library(faraway)` or `require(faraway)`
- Entfernen aus einer laufenden Sitzung
  `detach(package:faraway)`

Mathematische Operatoren

Symbol	Funktion
^ oder **	Potenz
*, /, +, -	Multiplikation, Division, Addition, Subtraktion
%/%	ganzzahlige Division
%%	modulo Division
%*%	Matrixmultiplikation

Natürlich gibt es alle üblichen mathematischen Operationen: `round()`, `sin()`, `abs()`, `sqrt()` etc. Wichtig für das Konzeptverständnis: Alle diese Operatoren sind gewöhnliche R-Funktionen:

```
> "+"(3,4)
[1] 7
```

Mathematische Operatoren

- Wichtig sind die Bezeichner für die speziellen Zahlen:
  - NaN : Not a Number,
  - Inf, -Inf : plus resp. minus unendlich,
  - NULL : nichts, leer,
  - TRUE, FALSE : Wahr oder falsch,
  - NA : not available, fehlender Wert, *missing value*.
- Achtung: R implementiert IEEE Arithmetik! Internationaler Standard.

```
> round(1.5) ; round(0.5)
[1] 2
[1] 0
```

- Achtung: `pi` ist nicht `PI`! R beachtet Groß- und Kleinschreibung!

Logische Operatoren

- == : als numerischer Vergleich: beide Objekt sind **identisch**,
- all.equal() testet auf numerische Gleichheit bis auf eine festgelegte Abweichung,
- `identical()` für Vergleich beliebiger Objekte,
- != : ungleich,
- <, > , <=, >= kleiner als, größer als (oder gleich),
- &, |, ! : (logisch) AND, OR, NOT .

Kleine Fallstricke

```
> a <- 3
> b <- 2.1/0.7
> a == b
[1] FALSE
```

Was passiert hier?

Kleine Fallstricke

```
> a <- 3
> b <- 2.1/0.7
> a == b
[1] FALSE
```

Was passiert hier? Als Gleitkommazahlen sind 3 und 2.1/0.7 im Rechner nicht identisch! Lösung in R: es gibt die Funktion `all.equal()`

```
> all.equal(a, b)
[1] TRUE
> ?all.equal
```

`all.equal()` überprüft die numerische Gleichheit bis auf ein ϵ
Standard: `sqrt( .Machine.double.eps )`

Kleine Fallstricke

Naiver Weise vermutet man, dass das Folgende funktioniert:

```
> a <- NA
> a == NA
```

oder

```
> a <- NaN
> a == NaN
```

Kleine Fallstricke

Naiver Weise vermutet man, dass das Folgende funktioniert:

```
> a <- NA
> a == NA

[1] NA
> a <- NaN
> a == NaN
[1] NA
```

Macht es aber nicht!
Für diese Fälle stellt R Folgendes zur Verfügung:

```
> a <- NA ; is.na(a)
> a <- NaN ; is.nan(a)
```

Elementare Statistik (Statistik I)

Vielzahl eingebauter Funktionen!

- `mean()`, `var()`, `sd()`, `cor()` etc.
- `runif()`, `rnorm()` etc. Zufallszahlenerzeugung
- `fivenum()`, `range()`, `summary()`, `stem()` Tukey's numbers, Spannweite, Stem-and-leaf plot
- `boxplot()`, `pie()`, `hist()`, `barplot()` grundlegende grafische Darstellungen
- `lm()`, `t.test()` lineare Regression, t-Test

Umgang mit fehlenden Werten

- Die wichtige Option `'na.rm'` legt fest, wie NAs in Berechungen behandelt werden sollen.
  Insbesondere wichtig in der Form z.B. `mean(x, na.rm=TRUE)`, auch als globale Option `na.action`.

Kurzer Eindruck der Datenanalyse

```
> data(women)
> help(women)
> plot(women, xlab = "Height (in)", ylab = "Weight (lb)",
         main = "women data: American women aged 30-39")
> pdf(file="women.pdf")
> plot(women, xlab = "Height (in)", ylab = "Weight (lb)",
         main = "women data: American women aged 30-39")
> dev.off()
> names(women)
> lineareRegression <- lm(women$weight ~ women$height)
> summary(lineareRegression)
```

Kurzer Eindruck der Datenanalyse

```
> data(iris)
> names(iris)
> str(iris)
> ?iris
> summary(iris)
> attach(iris)
> species.n <- as.numeric(Species)
> plot(iris, col=species.n)
> hist(Petal.Length)
> op <- par(mfrow=c(2,2))
> for (i in 1:4){
  boxplot(iris[,i] ~ Species, main = colnames(iris[i]))}
> par(op)
```

Und das ganze in lesbar: Literate Programming

- Der Begriff wurde bereits vor 30 Jahren in einem Artikel von Donald Knuth geprägt. (Donald E. Knuth: Literate Programming. In: The Computer Journal. 27, Nr. 2, 1984, S. 97–111)
- Die Idee ist, beschreibenden Text und das eigentlich Computerprogramm in einem lesbaren(!) Dokument zu vereinen und bei Bedarf daraus entweder einen Text oder ein Programm zu extrahieren.
- Unmittelbarer Vorteil: Das Dokument ist auch Jahre später noch selbsterklärend.
- `RSTudio` zusammen mit `knitr` bieten optimale Vorraussetzungen dieses Konzept in `R` umzusetzen.

Literate Programming am Beispiel

- Vorraussetzung: Die Bibliothek `knitr` muss geladen werden: `library(knitr)`
- Neue Datei des Typs R Markdown öffen.
- Markdown ist eine Auszeichungssprache wie HTML oder LaTeX, aber abgespeckt!
- Der in `knitr` implementierte Dialekt ist unter `http://www.rstudio.com/ide/docs/authoring/using_markdown` dokumentiert.
- Im Beispieltext sind die Auszeichnungen für Überschriften, R Chunks, Fettdruck und das Einbetten von Plots zu sehen.
- Einfügen des Programmtexts aus dem Beispiel `data(women)`
- Abspeichern als z.B. `women.Rmd` im Arbeitsverzeichnis.

Die women.Rmd Datei

````
Gewicht und Größe eine Population weiblicher homo sapiens
==========================================================
Zunächst muss der Datensatz eingelesen werden:
```{r}
data(women)
```
Über das Kommando 'data' werden mitgelieferte Datensätze
in eine R Sitzung geladen.

Über viele Datensätze gibt ausführliche Informationen,
die als Hilfeseiten mitgeliefert werden.
```{r}
help(women)
```
````

Einige Kommandos lassen den Datensatz genauer erkunden.

```{r}
names(women)
str(women)
```

Auch Bilder lassen sich in ein solches Dokument einbetten.
Als Beispiel soll der Zusammenhang zwischen Größe und Gewicht der Individuen angeschaut werden.

```{r}
plot(women, xlab = "Height (in)", ylab = "Weight (lb)",
       main = "women data: American women aged 30-39")
```

Das Bild lässt einen linearen Zusammenhang vermuten.
Das lässt sich leicht mit einer Regression überprüfen.

```{r}
lineareRegression <- lm(women$weight ~ women$height)
summary(lineareRegression)
```

Die Variable 'lineareRegression' enthält das vollständige Ergebnis einer Regression in R. Einige Kommandos wissen, wie mit Objekten verschiedenen Typs umzugehen ist, z.B. 'plot'.

```{r}
plot(lineareRegression)
```

Ist die Regression angemessen? Na ja.

Jedenfalls bekommt man mit *.Rmd Dokumenten eine Möglichkeit leicht lesbare Dokumente zu erzeugen.

Und hier das Ergebnis im Netz.

Aufgabe

Bitte lesen Sie die Dokumente:

- http://de.wikipedia.org/wiki/Markdown
- http://rmarkdown.rstudio.com/lesson-1.html bis
- http://rmarkdown.rstudio.com/lesson-10.html

Die grundlegenden Konzepte von R

- R ist ein klassischer Interpreter, der in einer sogenannten *read-eval-print-loop* arbeitet.
- Es wird Zeile für Zeile eingelesen, jeweils bis der Interpreter das Ende eines Codeblocks erkennt. Das Einlesen kann auch direkt aus einer Datei geschehen! (siehe `source()`)
- Jede Evaluation geschieht auf genau einer sog. R expression (siehe `eval(EXPR)`).
- Funktionen sind keine speziellen Sprachelemente, sondern einfache Objekte. Benutzerdefinierte Funktionen sind sehr leicht hinzuzufügen (lexical scoping!):

```
> datdobbelde <- function(x) {invisible(2*x)}
> x<-2
> datdobbelde(x)
### Keine Ausgabe!
> (datdobbelde(x)) # entspricht print(datdobbelde(x))
```

Atomare Datentypen in R

Datentyp	Beispiel
NULL	NULL
logical	FALSE
numeric	3.14
complex	3+i
character	"Hello"
factor	Ford, GM, Mercedes

- Zahlen haben einen `mode` und einen `type`
- Man findet den Typ eines Objektes mit `is.logical()`, `is.numeric()` etc. heraus.
- Man kann in RTypumwandlung erzwingen durch `as.factor()`, `as.numeric()` etc.

Zusammengesetzte Datentypen in R

- R ist eine vektororientierte Sprache. Alle komplexeren Datentypen sind aus sogenannten generischen Vektoren zusammengesetzt.
- Der wichtigste Datentyp für Berechnungen ist der `vector`, eine spezielle indizierte Liste von Elementen des selben Typs.
- Ein Vektor besteht (fast) immer aus Elementen eines Typs! Wenn bei der Konstruktion eines Vektors verschiedene Typen zusammengefasst werden, werden diese automatisch auf den einfachsten möglichen gemeinsamen Typen konvertiert!
- Einzige Ausnahme ist die Liste, ein spezieller Vektor, der aber beliebige Einträge haben kann.
- Das elementare Kommande heißt `c()` (**c**`ombine`).
- Skalare sind Vektoren der Länge 1.

Umgang mit Vektoren (Erzeugung)

```
> (x <- c(1, 3, 4.5))
[1] 1.0 3.0 4.5
> typeof(x)
[1] "double"
> (x <- c(1, x, 3))
[1] 1.0 1.0 3.0 4.5 3.0
> length(x) ### length of vector
[1] 3
> x <- c(1, 4, "Hello")
> t(x)      ### transposing (vectors / matrices)
[1,] "1" "4" "Hello"
```

Umgang mit Vektoren (Teilmengenauswahl)

```
> (x <- c(2, 3, 5, 7, 11, 13, 17, 19, 23))
[1]  2  3  5  7 11 13 17 19 23
> x[1]            ### Zugriff über den einfachen Index
[1] 2
> x[2:4]          ### mehrere Elemente auf einmal
[1] 3 5 7
> x[-(2:4)]       ### einige auslassen
[1]  2 11 13 17 19 23
> x[-c(1, 7, 9)] ## Indizes müssen nicht aufeinander folgen
[1]  3  5  7 11 13 19
> x[]             ### der komplette Vektor
[1]  2  3  5  7 11 13 17 19 23
```

Umgang mit Vektoren (bedingte Teilmengen)

```
> which(x < 10)
[1] 1 2 3 4
### Indizes, für die eine Bedingung erfüllen
> x
[1]  2  3  5  7 11 13 17 19 23
> x > 10
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> sum(x>10)
5
### Anzahl der Elemente, die eine Bedingung erfüllen
> x [ x > 10 ] # Indiziern über booleschen Vektor
[1] 11 13 17 19 23
### Äquivalent ist subset()
> subset(x, x<15)
[1] 2  3  5  7 11 13
```

Erzeugung spezieller Vektoren seq(), rep()

```
> 1:5; 5:1      ### äquidistant, Distanz 1
[1] 1 2 3 4 5
[1] 5 4 3 2 1

> seq(1,4,2/3) ### äquidistant, Distanz != 1
[1] 1.000000 1.666667 2.333333 3.000000 3.666667

> seq(along=x) ### Numerierung von x
[1] 1 2 3 4 5 6 7 8 9

> rep(TRUE, 5) ### Wiederholungen
[1] TRUE TRUE TRUE TRUE TRUE

> rep(c("red","blue"),c(4,7))
[1] "red"  "red"  "red"  "red"  "blue" "blue" "blue" "blue"
[9] "blue" "blue" "blue"
```

Rechnen mit R-vektoren

```
> (x <- seq(3,7))
[1] 3 4 5 6 7

> 1+x    ### Recycling der 1
[1] 4 5 6 7 8

> 2*x    ### elementweise Berechnung
[1]  6  8 10 12 14

> x*x    ### elementweise Multiplication von Vektoren
[1]  9 16 25 36 49

> x%*%x  ### Skalarproduct, implizite Transposition!
     [,1]
[1,]  135
```

Vektorrecycling

```
> 6:1 - 1:3 ### der kürzere Vektor wird recycelt
[1]  5  3  1  2  0 -2

> 7:1 - 1:3 ### Nur, wenn man weiß. was man tut!
[1]  6  4  2  3  1 -1  0
Warning message:
longer object length
is not a multiple of shorter object length in: 7:1 - 1:3
```

Wertetabellen

- R kann oft Schleifen vermeiden. Tun Sie es!
- Nicht im Geiste von R:

```
res <- matrix(ncol=10, nrow=10)
for (i in 1:10) for (j in 1:10) res[i,j] <- i*j
```

- The R Way (much, much faster)

```
res <- outer(1:10, 1:10, "*")
```

- Wie viel schneller? Faktor 1000!

```
> res <- matrix(ncol=1000, nrow=1000) ; system.time (
  for (i in 1:1000) for (j in 1:1000) res[i,j] <- i*j)
     User      System verstrichen
    2.542       0.009       2.564
> system.time(res <- outer(1:1000, 1:1000, "*"))
     User      System verstrichen
    0.002       0.008       0.010
```

Eingebautes Hilfesystem

- Das cheat-sheet für R:
  `http://cran.r-project.org/doc/contrib/Short-refcard.pdf`
- `help` oder "?" sind äquivalent zu RTFM: Versuchen Sie `help(plot)` oder `?plot`.
- Wenn man das genaue Kommando nicht weiß oder `help()` nicht hilft, dann kann man `apropos()`, `find()` oder `help.search()` versuchen.
- Versteht man eine Hilfeseite nicht, dann kann man mit `example(`*command*`)` oder `demo(`*command*`)` versuchen, den Befehl und seine Nutzung am Beispiel zu lernen.

Eingebautes Hilfesystem

- `help.start()` zeigt die Dokumentation im Standard-Webbrowser an.
- Die meisten von Nutzern hinzugefügten Pakete enthalten eine sog. Vignette, eine kurzes Handbuch im PDF Format. Mit dem Kommando `vignette()` kann man sich dieses anzeigen lassen.

Externe Hilfe

- Dokumentation auf CRAN: cran.r-project.org
  Sehr viel gut geschriebene Dokumentation!
  Installationshandbuch, Referenzhandbuch, Dokumentation für Datenaustausch, **FAQ** usw.
- Archive der Mailinglisten mit Suchinterface auf CRAN
  `http://cran.r-project.org/search.html`

Ultima ratio

- Selbst auf der Mailingliste r-help fragen. Unbedingt den posting guide beachten, sonst wird man ge'ripleyed'. Mehrere tausend Leser, mehr als 100 Mails am Tag. Es gibt praktisch auf jede vernünftig gestellte Frage ein fundierte Antwort.
- Bekommt man sein Problem gelöst, so sollte man sein Wissen teilen, in dem man die Antwort z.B. in das R-Wiki `http://wiki.r-project.org/rwiki/doku.php` einträgt.
- Es gibt auch eine eigene Gruppe auf `stackoverflow.com`: `http://stackoverflow.com/questions/tagged/r`