R Reference Card

von Jörg Gersonde, 2011-06-17
(nach der *R Reference Card* von Tom Short mit freundlicher Erlaubnis)

Hilfe und Grundlagen

Die meisten R-Funktionen haben eine online-Dokumentation.

help(topic) Hilfe zu topic
?topic ebenso.
help.search("topic") durchsuche die Hilfetexte
apropos("topic") die Namen aller gefundenen Objekte, auf die der reguläre Ausdruck "topic" zutrifft
help.start() starte die HTML-Version der Hilfe
str(a) zeige die interne *Str*uktur eines R-Objektes
summary(a) gibt ein "summary" von a, oft eine statistische Übersicht, *generische* Funktion: für verschiedene Klassen von a wird ein anderes Ergebnis geliefert
ls() zeigt die Objekte im Suchpfad; spezifiziere pat="pat" für die Suche nach Mustern.
ls.str() str() für jede Variable im Suchpfad
dir() zeigt alle Dateien im aktuellen Verzeichnis
methods(a) zeigt alle S3-Methoden von a
methods(class=class(a)) listet alle verfügbaren Methoden für ein Objekt der Klasse a
options(...) setzt oder zeigt viele globale Optionen; häufig: width, digits, error
library(x) lädt add-on packages; library(help=x) listet Datasets und Funktionen in package x.
attach(x) fügt x zum R-Suchpfad hinzu; x kann sein: eine Liste, ein Dataframe oder R-Datendatei, welche mit save erzeugt wurde. Verwende search() zur Anzeige des Suchpfades.
detach(x) entferne x aus dem R-Suchpfad; x ist der Name eines Objektes, das vorher zum Suchpfad hinzugefügt wurde, oder der Name eines package.

Ein- und Ausgabe

load() lade Variable, die mit save geschrieben wurden.
data(x) lade vordefinierten Dataframe
read.table(file) lese ASCII-Datei ein und erzeuge einen Dataframe; Standardfeldtrenner sep="" sind beliebige whitespace; header=TRUE erste Zeile enthält Spaltennamen; as.is=TRUE Zeichenkettenvektoren nicht in Faktoren umwandeln; comment.char="" nicht "#" als Kommentarzeichen interpretieren; skip=n n Zeilen überspringen vor dem Lesen; siehe Hilfetext für weitere Optionen
read.csv("filename",header=TRUE) ebenso, aber mit Parametern zum Einlesen von CSV-Dateien
read.delim("filename",header=TRUE) ebenso, aber mit Tabulator als Trennzeichen
read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE) lese Tabelle, deren Daten mit *f*ixed *w*idth *f*ormatiert sind; über den integer-Vektor widths werden die Spaltenbreiten der Felder vorgegeben.
save(file,...) speichert die Objekte (...) im XDR-Format in der Datei *file*, plattformunabhängig
save.image(file) speichert alle Objekte
cat(..., file="", sep=" ") "druckt" alle Argumente nach Umwandlung in Zeichenketten; sep ist das Trennzeichen
print(a, ...) druckt die Argumente; generisch: Ausgabe abhängig von der Objektklasse
format(x,...) hübsches Drucken eines R-Objektes (mehr Optionen)
write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") druckt x nach der Umwandlung in einen Dataframe; wenn quote TRUE, dann werden Werte von Spalten mit Zeichenketten oder Faktoren von Anführungszeichen (") umschlossen; sep ist der Feldtrenner; eol ist das Zeilenendezeichen; na ist die Zeichenkette für fehlende Werte; verwende col.names=NA, um die Kopfzeile für das Einlesen in Tabellenkalkulationsprogramme anzupassen
sink(file) sende Ausgabe in die Datei file bis sink()

Die meisten I/O-Funktionen haben einen Parameter file. Dieser kann der Name einer Datei oder Verbindung (z.B. URL) sein. file="" bedeutet Standardinput oder -output. Verbindungen können sein: Dateien, pipes, gezippte Dateien und R-Variablen.

Unter Windows, kann die Verbindung auch description = "clipboard" verwendet werden. Um eine Tabelle von Excel zu kopieren, verwende
x <- read.delim("clipboard")
Um für Excel eine Tabelle in die Zwischenablage zu schreiben, verwende
write.table(x,"clipboard",sep="\t",col.names=NA)
Für die Nutzung von Datenbanken, siehe die Pakete RODBC, DBI, RMySQL, RPgSQL, und ROracle. Siehe die Pakete foreign, XML, hdf5, netCDF zum Lesen anderer Formate.

Daten erzeugen

c(...) generische Funktion, um die Argumente zu kombinieren; erzeugt standardmäßig einen Vektor; mit recursive=TRUE Absteigen in Listen, um alle Elemente in einen Vektor zu packen
from:to generiert Folgen; ":" Operator (Vorrang beachten); 1:4 + 1 is "2,3,4,5"
seq(from,to) generiert eine Sequenz by= spezifiziert Schrittweite; length= spezifiziert Länge der Sequenz
seq(along=x) generiert 1, 2, ..., length(x); nützlich für for Schleifen
rep(x,times) wiederholt x times; nutze each= um jedes Element von x each-mal zu wiederholen; rep(c(1,2,3),2) ist 1 2 3 1 2 3; rep(c(1,2,3),each=2) ist 1 1 2 2 3 3
data.frame(...) erzeugt einen Dataframe aus benannten oder unbenannten Argumenten;
data.frame(v=1:4,ch=c("a","B","c","d"),n=10); kürzere Vektoren werden recycled bis zur Länge des längsten
list(...) erzeugt eine Liste aus benannten oder unbenannten Argumenten; list(a=c(1,2),b="hi",c=3i);
array(x,dim=) Array mit Daten x; spezifiziere die Dimensions: dim=c(3,4,2); Elemente von x werden recycled, falls x nicht lang genug ist
matrix(x,nrow=,ncol=) Matrix; Elemente von x werden recycled
factor(x,levels=) kodiert einen Vektor x als Faktor
gl(n,k,length=n*k,labels=1:n) erzeugt Levels (factors) durch Angabe der Muster der Levels; k ist die Anzahl der Levels, und n ist die Anzahl der Wiederholungen
expand.grid() ein Dataframe aus allen Kombinationen der enthaltenen Vektoren oder Faktoren
rbind(...) kombiniert die Argumente als Zeilen von Matrizen Dataframes u.a.
cbind(...) stellt die Argumente spaltenweise nebeneinander

Zugriff auf Datenelemente, -teile

Indizierung von Listen

x[n]	Liste mit den Elementen n
x[[n]]	n^{tes} Element der Liste
x[["name"]]	Element der Liste mit dem Namen "name"
x$name	id.

Indizierung von Vektoren

x[n]	n^{tes} Element
x[-n]	alle *außer* das n^{te} Element
x[1:n]	die ersten n Elemente
x[-(1:n)]	Elemente ab n+1 bis zum Ende
x[c(1,4,2)]	einzelne Elemente
x["name"]	Element mit dem Namen "name"
x[x > 3]	alle Elemente größer als 3
x[x > 3 & x < 5]	alle Elemente zwischen 3 und 5
x[x %in% c("a","und","das")]	Elemente in gegebener Menge

Indizierung von Matrizen

x[i,j]	Element bei Zeile i, Spalte j
x[i,]	i-te Zeile
x[,j]	j-te Spalte
x[,c(1,3)]	Spalten 1 und 3
x["name",]	die Zeile mit dem Namen "name"

Indizierung von Dataframes (Matrizenindizierung plus folgende)
Dataframes sind spezielle Listen.

x[["name"]]	Spalte mit dem Namen "name"
x$name	id.
x["name"]	id. aber als Dataframe

S4-Objeke
x@name Zugriff auf Slot "name" von "x"

Variablen konvertieren

as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.complex(x), as.character(x), ... Typkonvertierung; siehe methods(as)

Variablen-Information

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ... testet auf einen bestimmten Typ; siehe methods(is)
length(x) Anzahl der Elemente in x
dim(x) anzeigen/setzen der Dimension eines Objektes; dim(x) <- c(3,2)
dimnames(x) anzeigen/setzen der Dimensionsnamen eines Objektes
nrow(x) Anzahl der Zeilen; NROW(x) genauso aber behandelt einen Vektor als einspaltige Matrix
ncol(x) and **NCOL(x)** id. für Spalten
class(x) anzeigen/setzen der Klasse von x; class(x) <- "myclass"
unclass(x) entferne das Klassenattribut von x

attr(x,which) anzeigen/setzen des Attributes `which` von `x`
attributes(obj) anzeigen/setzen der Liste der Attribute von `obj`

Datenselektion und -manipulation

which.max(x) Index des größten Elementes von **x**
which.min(x) Index des kleinsten Elementes von **x**
rev(x) kehrt die Reihenfolge der Elemente von `x` um
sort(x) sortiert die Elemente von `x` aufsteigend; für absteigend: `rev(sort(x))`
cut(x,breaks) teilt `x` in Intervalle auf (factors); `breaks` ist die Anzahl der Intervalle oder ein Vektor mit den Intervallgrenzen
match(x, y) gibt einen Vektor zurück mit der gleichen Länge wie `x` mit den Elementen von `x`, welche auch in `y` sind (`NA` sonst)
which(x == a) ergibt einen Vektor mit den Indizes `x`, für die der Vergleich wahr (`TRUE`) ist, in diesem Beispiel die Werte `i`, für welche `x[i] == a` (das Argument dieser Funktion muss eine Variable mit dem Mode `logical` sein)
choose(n, k) berechnet die Anzahl der Kombinationen von k Elementen aus n ($= n!/[(n-k)!k!]$)
na.omit(x) unterdrücke die Beobachtungen mit fehlenden Werten (`NA`) (unterdrücke die entsprechende Zeile, wenn `x` eine Matrix oder ein Dataframe ist)
na.fail(x) liefert eine Fehlermeldung, wenn `x` mindestens ein `NA` enthält
unique(x) wenn `x` ein Vektor oder eine Matrix ist, ein ähnliches Objekt aber ohne doppelte Elemente
table(x) liefert eine Tabelle mit Anzahlen der verschiedenen Werte von `x` (typisch für integers oder factors)
subset(x, ...) liefert eine Auswahl von `x` entsprechend der Kriterien (`...`, typische Vergleiche: `x$V1 < 10`); ist `x` ein Dataframe, dann gibt der Parameter `select` an, welche Variablen (Spalten) erhalten bleiben bzw. (bei einem Minuszeichen) entfernt werden.
sample(x, size) erzeugt ein Sample von `x` ohne Zurücklegen mit `size` Elementen, verwende `replace = TRUE` für Sample mit Zurücklegen
prop.table(x,margin=) Tabellenwerte als Anteile der Randsummen

Math

sin,cos,tan,asin,acos,atan,atan2,log,log10,exp
max(x) Maximum der Elemente von `x`
min(x) Minimum der Elemente von `x`
range(x) id. wie `c(min(x), max(x))`
sum(x) Summe der Elemente von `x`
diff(x) Differenzenberechnung für den Vektor `x`
prod(x) Produkt der Elemente von `x`
mean(x) arithm. Mittel der Elemente von `x`
median(x) Median der Elemente von `x`
quantile(x,probs=) bestimme Quantile zu den gegebenen Wahrscheinlichkeiten (defaults to 0,.25,.5,.75,1)
weighted.mean(x, w) gewogenes arithm. Mittel von `x` mit Gewichten `w`
rank(x) Rang der Elemente von `x`
var(x) oder `cov(x)` Varianz der Elemente von `x` (berechnet mit $n-1$); wenn `x` eine Matrix oder ein Dataframe ist, wird die Varianz-Kovarianz-Matrix berechnet.
sd(x) Standardabweichung von `x`
cor(x) Korrelationmatrix von `x` (Matrix oder Dataframe); sonst 1 (`x` ist ein Vektor)
var(x, y) oder **cov(x, y)** Kovarianz zwischen `x` und `y`, oder zwischen den Spalten von `x` und denen von `y`, wenn es Matrizen oder Dataframes sind.
cor(x, y) lineare Korrelation zwischen `x` und `y`, oder Korrelationmatrix bei Matrizen und Dataframes.
round(x, n) rundet die Elemente von `x` auf `n` Stellen
log(x, base) Logarithmus von `x` zur Basis `base`
scale(x) wenn `x` eine Matrix ist, zentrieren und skalieren der Daten; Optionen: `scale=FALSE`, `center=FALSE` (Standard: `center=TRUE`, `scale=TRUE`)
pmin(x,y,...) ein Vektor, dessen *i*tes Element das Minimum ist von `x[i]`, `y[i]`, ...
pmax(x,y,...) id. für Maximum
cumsum(x) ein Vektor, dessen *i*tes Element die Summe ist von `x[1]` bis `x[i]`
cumprod(x) id. für das Produkt
cummin(x) id. für das Minimum
cummax(x) id. für das Maximum
union(x,y), **intersect(x,y)**, **setdiff(x,y)**, **setequal(x,y)**, **is.element(el,set)** "Mengen"-Funktionen
Re(x) Realteil einer komplexen Zahl (aus $\mathbb{C}$)
Im(x) Imaginärteil von `x`
Mod(x) modulus; `abs(x)` ist das gleiche
Arg(x) Winkel (in Bogenmaß) einer komplexen Zahl
Conj(x) konjugiert komplexe Zahl
convolve(x,y) berechnet verschiedene Arten Convolution von zwei Reihen
fft(x) Fast Fourier Transform eines Arrays
mvfft(x) FFT für jede Spalte eine Matrix
filter(x,filter) wendet einen linearen filter auf eine univariate Zeitreihe an oder auf jeder Reihe (bei multivariaten Zeitreihen)

Viele mathematische Funktionen haben den Parameter `na.rm=FALSE` für die Behandlung fehlender Werte (NA).

Matrizen

t(x) transponiere
diag(x) setzt/extrahiert Diagonalelemente; konstruiert Diagonalmatrix
%*% Matrizenmultiplikation
solve(a,b) löst `a %*% x = b` nach `x` auf
solve(a) inverse Matrix von `a`
rowsum(x) Zeilensummen für ein Matrix-ähnliches Objekt; **rowSums(x)** id. - nur schneller
colsum(x), **colSums(x)** id. für Spalten
rowMeans(x) Mittelwerte für Zeilen.
colMeans(x) id. für Spalten

Höhere Datenverarbeitung

apply(X,MARGIN,FUN=) Vektor, Matrix oder Liste mit den Werten, die durch Anwendung von Funktion `FUN` auf die Zeilen (Spalten) (`MARGIN`=1 (2)) von `X` entstehen
lapply(X,FUN) wendet `FUN` auf jedes Element der Lista `X` an
tapply(X,INDEX,FUN=) wendet `FUN` auf jede Gruppe von Werten von `X` an bei Gruppierungsschlüssel `INDEX`
by(data,INDEX,FUN) wendet `FUN` auf den Dataframe `data` an mit Teilmengen laut `INDEX`
ave(x,...,FUN=mean) Untermengen von `x` werden gemittelt (oder andere Funktion `FUN`), die Untermengen werden durch die Faktorvariablen ... definiert
merge(a,b) mischt zwei Dataframes entsprechend gemeinsamer Spaltennamen
xtabs(a b,data=x) Kontingenztabelle bezüglich zweier Faktoren
aggregate(x,by,FUN) spaltet den Dataframe `x` in Teilmengen, berechne für jede die zusammenfassende Statistik entsprechend `FUN` und gibt das Ergebnis in entsprechender Form zurück; `by` ist die Liste der gruppierenden Elements - jedes so lang wie `x`
stack(x, ...) transformiert die Daten in eine einzelne Spalte.
unstack(x, ...) das Gegenteil von `stack()`
reshape(x, ...) formt einen Dataframe um zwischen 'breitem' Format (mit wiederholten Messungen in verschiedenen Spalten des gleichen Datensatzes) und dem 'langen' Format (mit Wiederholungen in jeweils verschiedenen Datensätzen); (direction="wide") oder (direction="long")

Zeichenketten

paste(...) fügt Vektoren zusammen (nach Umwandlung zu Zeichenketten); `sep=` ist der einzufügende Trenner (einzelnes Leerzeichen ist Standard); `collapse=` optionales Trennzeichen für "kollabierte" Ergebnisse
substr(x,start,stop) Teilzeichenketten finden; auch als Zuweisung: `substr(x, start, stop) <- value`
strsplit(x,split) spaltet `x` bei Auftreten von Zeichenkette `split`
grep(pattern,x) sucht nach Treffern `pattern` in `x`; siehe `?regex`
gsub(pattern,replacement,x) ersetzt alle Treffer (regulärer Ausdruck) `sub()` ersetzt nur den ersten Treffer.
tolower(x) konvertiert zu Kleinbuchstaben
toupper(x) konvertiert zu Großbuchstaben
match(x,table) ein Vektor mit den Positionen des jeweils ersten Treffers der Elemente von `x` in `table`
x %in% table id. aber gibt einen Vektor mit `TRUE`/`FALSE`
pmatch(x,table) partielle Treffer von Elementen aus `x` in `table`
nchar(x) Anzahl der Zeichen

Datum und Zeit

Die Klasse `Date` behandelt Datum ohne Zeit. `POSIXct` für Datum und Zeit inkl. Zeitzone. Vergleiche (z.B. >), `seq()`, und `difftime()` sind möglich. `Date` kann auch + and −. `?DateTimeClasses` für weitere Informationen. Siehe auch Paket `chron`.

as.Date(s) und **as.POSIXct(s)** konvertieren in die entsprechende Klasse; `format(dt)` konvertiert zu einer Zeichenkette. Das Standardformat ist "2001-02-21". Über ein zweites Argument kann das Format festgelegt werden. Einige übliche Formate:

`%a`, `%A` Abkürzung, voller Wochentagsname.
`%b`, `%B` Abkürzung, voller Monatsname..
`%d` Tag im Monat (01–31).
`%H` Stunde (00–23).
`%I` Stunde (01–12).
`%j` Tag im Jahr (001–366).
`%m` Monat (01–12).
`%M` Minute (00–59).
`%p` AM/PM-Indikator.
`%S` Sekunde als Dezimalzahl (00–61).
`%U` Woche (00–53); erster Sonntag ist Tag 1 der Woche 1.
`%w` Wochentag (0–6, Sonntag is 0).
`%W` Woche (00–53); erster Mondtag ist Tag 1 der Woche 1.
`%y` Jahr ohne Hunderter (00–99). mglst. nicht verwenden.
`%Y` Jahr mit Jahrhundert.
`%z` (nur output.) Offset von Greenwich; `-0800` ist 8 Stunden westlich.
`%Z` (nur output.) Zeitzone als Zeichenkette (leer, wenn nicht verfügbar.

Führende Nullen werden ausgegeben, aber bei der Eingabe sind diese optional. Siehe `?strftime`.

Grafik ausgeben

x11(), windows(), quartz() öffne ein Grafikfenster
postscript(file) startet den Gerätetreiber zum Erstellen von PostScript-Grafiken; `horizontal = FALSE`, `onefile = FALSE`, `paper = "special"` für EPS-Dateien; `family=` spezifiziert die Schriftart (AvantGarde, Bookman, Courier, Helvetica, Helvetica-Narrow, NewCenturySchoolbook, Palatino, Times, or Computer-Modern); `width=` und `height=` Einstellen des Bereiches in Zoll (für `paper="special"`, definiert die Papiergröße).
ps.options() setzen und ansehen (Aufruf ohne Argumente) der Standardwerte der Argumente von `postscript`
pdf, png, jpeg, bitmap, xfig, pictex; siehe `?Devices`
dev.off() schließt das angegebene (standardmäßig das aktuelle) Grafikgerät; siehe auch `dev.cur`, `dev.set`

Diagramme plotten

plot(x) plottet die Werte von `x` (entlang der *y*-Achse) sortiert an der *x*-Achse
plot(x, y) bivariater Plot von `x` (auf der *x*-Achse) und `y` (auf der *y*-Achse); **plot(y ~ x)** ebenso
hist(x) Histogramm der Häufigkeiten von `x`; `horiz=FALSE` für vertikale Balken
barplot(x) Barplot der Werte von `x`; `horiz=FALSE` für vertikale Balken
dotchart(x) wenn `x` ein Dataframe ist, wird ein Cleveland-Punkteplot erzeugt (stacked plots line-by-line and column-by-column)
pie(x) Tortendiagramm
boxplot(x) "box-and-whiskers"-Plot
sunflowerplot(x, y) ähnlich wie `plot()` aber Punkte mit ähnlichen Koordinaten werden als Blüten mit Anzahl von Blütenblättern entsprechend der Anzahl der repräsentierten Punkte
stripplot(x) zeichnet die Werte von `x` auf einer Linie (Alternative zu `boxplot()` für kleine Datenmengen)
coplot(x~y | z) bivariater Plot von `x` und `y` für jeden Wert oder Intervall der Werte von `z`
interaction.plot (f1, f2, y) sind `f1` und `f2` Faktoren, dann werden die Mittelwerte von `y` (auf der *y*-Achse) gezeichnet entsprechend der Werte von `f1` (auf der *x*-axis) und von `f2` (verschiedene Kurven); die Option `fun` erlaubt eine deskriptive Statistik von `y` (standardmäßig `fun=mean`)
matplot(x,y) bivariater Plot der ersten Spalte von `x` *vs.* der ersten von `y`; der zweiten von `x` *vs.* der zweiten `y` usw.; ohne `y` werden die Spalten von `x` (ab der zweiten) gegen die erste Spalte geplottet
fourfoldplot(x) stellt den Zusammenhang zwischen zwei dichotomen Variablen dar (mit Vierteln von Kreisen) für zwei Populationen. (`x` muss ein Array sein mit `dim=c(2, 2, k)`, oder eine Matrix mit `dim=c(2, 2)` wenn $k = 1$)
assocplot(x) Cohen–Friendly-Graph zeigt die Abweichung vom Zustand "Unabhängigkeit" der Zeilen und Spalten in einer zweidimensionalen Kontingenztafel
mosaicplot(x) 'mosaic' Darstellung der Residuen einer log-linearen Regression einer Kontingenztafel
pairs(x) ist `x` eine Matrix oder ein Dataframe, dann werden alle möglichen bivariaten Plots gezeichnet zwischen den Spalten von `x`
plot.ts(x) ist `x` ein object der Klasse `"ts"`, dann wird `x` entlang der Zeit aufgetragen, `x` darf multivariat sein (aber zur selben Zeitachse)
ts.plot(x) id. aber wenn `x` multivariat ist; die Zeitreihen dürfen verschiedene Zeitpunkte haben, müssen aber gleiche Frequenz aufweisen.
qqnorm(x) Quantile von `x` im Vergleich zur erwarteten Normalverteilung
qqplot(x, y) Quantile von `y` vs. Quantile von `x`
contour(x, y, z) Höhenlinien (die Werte werden interpoliert für die Kurven), `x` und `y` müssen Vektoren sein und `z` ein Matrix, so dass `dim(z)=c(length(x), length(y))` (`x` und `y` können weggelassen werden)
filled.contour(x, y, z) id. aber mit gefüllten Flächen zwischen den Höhenlinien und einer Legende zu den Farben
image(x, y, z) id. aber mit Farben (wirkliche Daten werden geplottet)
persp(x, y, z) id. aber in Perspektive (wirkliche Daten werden geplottet)
stars(x) ist `x` eine Matrix oder ein Dataframe, wird eine Grafik mit Segmenten oder Sternen gezeichnet, wobei jede Zeile von `x` durch einen Stern dargestellt wird und die Spalten durch die Länge der Segmente
symbols(x, y, ...) zeichnet spezielle Symbole an den Koordinaten (`x`, `y`), Symbole (Kreise, Quadrate, Rechtecke, Sterne, Thermometer oder "boxplots") mit Größe, Farbe, ... entsprechend der zusätzlichen Argumente
termplot(mod.obj) zeichnet die (partiellen) Effekte eines Regressionsmodells (`mod.obj`)

Die folgenden Parameter sind bei den meisten plot-Funktionen verfügbar:

add=FALSE wenn `TRUE`, wird der vorherige Plot überzeichnet (falls vorhanden)
axes=TRUE wenn `FALSE`, werden weder die Achsen noch der Rahmen gezeichnet
type="p" spezifiziert den Typ des Plots, `"p"`: Punkte, `"l"`: Linien, `"b"`: mit linien verbundene Punkte, `"o"`: id. aber mit Linien über den Punkten, `"h"`: vertikale Linien, `"s"`: Stufen, der Wert gehört zur Spitze der vertikalen Linien, `"S"`: id. aber der Wert gehört zum untersten Punkt der vertikalen Linien
xlim=, ylim= setzt untere und obere Grenzen für die Achsen, z.B. `xlim=c(1, 10)` oder `xlim=range(x)`
xlab=, ylab= Beschriftung der Achsen, muss eine Variable mit `mode character` sein
main= Titel, muss eine Variable mit `mode character` sein
sub= Untertitel (kleine Schrift)

Low-level Zeichenkommandos

points(x, y) fügt Punkte hinzu (`type=` kann verwendet werden)
lines(x, y) id. aber Linien
text(x, y, labels, ...) fügt den Text `labels` bei den Koordinaten (`x`,`y`) ein; Bsp.: `plot(x, y, type="n"); text(x, y, namen)`
mtext(text, side=3, line=0, ...) fügt den Text `text` in den Rand `side` ein (siehe `axis()` unten); `line` spezifiziert die Zeile im Zeichenbereich
segments(x0, y0, x1, y1) zeichnet Linien von den Punkten (`x0`,`y0`) zu den Punkten (`x1`,`y1`)
arrows(x0, y0, x1, y1, angle= 30, code=2) id. aber mit Pfeilen bei den Punkten (`x0`,`y0`) (wenn `code=2`), bei (`x1`,`y1`) (wenn `code=1`) oder beide (wenn `code=3`); `angle` kontrolliert den Winkel vom Schaft des Pfeiles zu den Ecken des Pfeilkopfes
abline(a,b) zeichnet eine Linie mit dem Anstieg `b` und Abschnitt `a`
abline(h=y) zeichnet eine horizontale Linie bei Ordinate `y`
abline(v=x) zeichnet eine vertikale Linie bei Abzisse `x`
abline(lm.obj) zeichnet eine Regressionsgerade für `lm.obj`
rect(x1, y1, x2, y2) zeichnet Rechtecke mit den Koordinaten `x1`, `x2`, `y1` und `y2`
polygon(x, y) zeichnet ein Polygon, indem die Punkte mit den Koordinaten `x` und `y` miteinander verbunden werden
legend(x, y, legend) fügt eine Legende hinzu bei (`x`,`y`) mit den Symbolen aus `legend`
title() fügt einen Title (und optional einen Untertitle) hinzu
axis(side) fügt eine Achse hinzu: unten (`side=1`), (`2`), oben (`3`), rechts (`4`); `at=vect` (optional) gibt die Abzisse (oder Ordinate) für die tick-Marken an
box() zeichne eine Box um den aktuellen Plot
rug(x) zeichnet die Daten `x` auf der *x*-Achse als schmale vertikale Linien
locator(n, type="n", ...) gibt die Koordinaten (*x*,*y*) zurück, nachdem der Nutzer `n`-mal auf den Plot geklickt hat; zeichnet auch Symbole (`type="p"`) oder Linien (`type="l"`) entsprechend der optionalen Parameter (`...`); standardmäßig wird nichts gezeichnet (`type="n"`)

Grafikparameter

Können mit **par(...)** gesetzt werden; viele können auch als Parameter den plot-Kommandos mitgegeben werden.

adj Textbund (`0` linksbündig, `0.5` zentriert, `1` rechtsbündig)
bg Farbe des Hintergrundes (z.B.. : `bg="red"`, `bg="blue"`, ... Die Liste der 657 verfügbaren Farben kann mit `colors()` angezeigt werden)

bty Rahmentyp, mögliche Werte: `"o"`, `"l"`, `"7"`, `"c"`, `"u"` ou `"]"` (der Rahmen sieht dann etwa wie das Zeichen aus); wenn `bty="n"`, wird kein Rahmen gezeichnet

cex Größe von Text und Symbolen; die folgenden Parameter wirken entsprechend für: die Werte auf den Achsen `cex.axis`, Beschriftung der Achsen `cex.lab`, Titel `cex.main`, Untertitel `cex.sub`

col Farbe von Symbolen und Linien; Farbnamen: `"red"`, `"blue"` see `colors()` oder `"#RRGGBB"`; siehe `rgb()`, `hsv()`, `gray()`, und `rainbow()`; wie bei `cex` gibt es auch: `col.axis`, `col.lab`, `col.main`, `col.sub`

font Schriftart (`1`: normal, `2`: italic, `3`: fett, `4`: fett italic); entsprechend: `font.axis`, `font.lab`, `font.main`, `font.sub`

las Orientierung der Achsenbeschriftung (`0`: parallel zu den Achsen, `1`: horizontal, `2`: senkrecht zu den Achsen, `3`: vertikal)

lty Linientyp (Integer oder Zeichenkette): `1`: `"solid"`, `2`: `"dashed"`, `3`: `"dotted"`, `4`: `"dotdash"`, `5`: `"longdash"`, `6`: `"twodash"`, oder eine Zeichenkette mit bis zu acht Zeichen (von `"0"` bis `"9"`, welche das Linienmuster festlegt, z.B. `lty="44"` hat den gleichen Effekt wie `lty=2`

lwd Linienstärke, default `1`

mar Vektor mit 4 Werten für den Abstand zwischen den Achsen und dem Rand der Gafik `c(bottom, left, top, right)`, die Standardwerte sind `c(5.1, 4.1, 4.1, 2.1)`

mai id. Abstände in Inch (Zoll)

mfcol ein Vektor mit zwei Werten `c(nr,nc)` zur Unterteilung des Grafikfensters in `nr` Zeilen und `nc` Spalten, die Zeichnungen erfolgen spaltenweise

mfrow id. aber Zeichnungen zeilenweise

pch bestimmt das zu zeichnende Symbol, entweder Ganzzahl von 1 bis 25, oder ein einzelnes Zeichen in `""`

1 ○ 2 △ 3 + 4 × 5 ◇ 6 ▽ 7 ⊠ 8 ✳ 9 ⊕ 10 ⊕ 11 ⧖ 12 ⊞ 13 ⊠ 14 ◪ 15 ■
16 ● 17 ▲ 18 ◆ 19 ● 20 ● 21 ○ 22 □ 23 ◇ 24 △ 25 ▽ * ⋆ . · X X a a ? ?

ps Ganzzahl für die Größe in Punkten (pt) von Text und Symbolen

pty bestimmt den Typ des Zeichenbereiches, `"s"`: square, `"m"`: maximal

tck Länge der tick-marks an den Achsen als Anteil relativ zur Breite oder Höhe des Plots; wenn `tck=1`, wird ein Gitter gezeichnet

tcl Länge der tick-marks an den Achsen als Anteil relativ zur Höhe einer Textzeile (Standard: `tcl=-0.5`)

xaxs, yaxs Stil für den Achsenabstand; Standard: `"r"` für zusätzlichen Abstand; `"i"` kein Abstand

xaxt wenn `xaxt="n"`, wird die *x*-Achse berechnet, aber nicht gezeichnet (nützlich zusammen mit `axis(side=1, ...)`)

yaxt id. für *y*-Achse (nützlich zusammen mit `axis(side=2, ...)`)

Lattice- (Trellis-) Grafiken

xyplot(y~x) bivariate Plots (mit vielen Optionen)

barchart(y~x) Histogramm von `y` in Abhängigkeit von `x`

dotplot(y~x) Cleveland-Punkte-Plot (gestapelte Plots Linie-für-Linie und Spalte-für-Spalte)

densityplot(~x) Dichtefunktionen-Plot

histogram(~x) Histogramm der Häufigkeiten von `x`

bwplot(y~x) "Box-and-Whiskers"-Plot

qqmath(~x) Quantile von `x` bei einer angenommenen theoretischen Verteilung der Werte

stripplot(y~x) eindimensionaler Plot, `x` numerisch, `y` kann ein Faktor sein

qq(y~x) Quantile für den Vergleich von zwei Verteilungen, `x` muss numerisch sein, `y` kann numerisch, character, oder ein Faktor sein, muss aber mindestens zwei 'levels' haben

splom(~x) Matrix mit bivariaten Plots

parallel(~x) Parallelkoordinaten-Plot

levelplot(z~x*y|g1*g2) farbiger Plot der Werte von `z` an den Koordinaten `x` und `y` (`x`, `y` und `z` müssen die gleiche Länge haben)

wireframe(z~x*y|g1*g2) 3d-Oberflächen-Plot

cloud(z~x*y|g1*g2) 3d-Streudiagramm

In der normalen Lattice-Formel, `y x|g1*g2` hat Kombinationen von optionalen konditionalen Variablen `g1` und `g2`, die in separaten Bereichen gezeichnet werden. Lattice-Funktionen haben meistens die Parameter der Basisgrafiken plus `data=` für den Dataframe und `subset=` für Teilmengen. Verwende `panel=` für die Definition einer eigenen panel-Funktion (siehe `apropos("panel")` und `?llines`). Lattice-Funktionen geben ein Object der Klasse trellis zurück und müssen mit `print()` behandelt werden zum Erzeugen der Grafik. Verwende `print(xyplot(...))` innerhalb von Funktionen, wo die automatische Ausgabe nicht erfolgt. Mit `lattice.theme` und `lset` werden die Standardwerte von Lattice verändert.

Optimierung und Schätzung von Modellen

optim(par, fn, method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN") allgemeine Optimierungsfunktion; `par` ist der Vektor der Startwerte, `fn` ist die zu optimierende Funktion (Standard: Minimierung)

nlm(f,p) minimiere Funktion `f` mit einem Algorithmus vom Newton-Typ mit den Startwerten `p`

lm(formula) schätze lineares Modell; `formula` hat typischerweise die Form `response termA + termB + ...`; setze `I(x*y) + I(x^2)` für nichtlineare Terme

glm(formula,family=) schätze ein verallgemeinertes lineares Modell, `formula` spezifiziert den linearen Prediktor und `family` beschreibt die Verteilung der Residuen und die link-Funktion des Modells; see `?family`

nls(formula) Least-Squares-Schätzer für ein Modell mit nichtlinearen Parametern

approx(x,y=) lineare Interpolation gegebener Datenpunkte; `x` kann Struktur wie bei xy-Plotting haben

spline(x,y=) Cubic-Spline-Interpolation

loess(formula) bestimme polynomiale Oberfläche per local fitting

Viele der formula-basierten Modellierungsfunktionen haben die gleichen Argumente: `data=` der Dataframe mit den Variablen für die Formel, `subset=` definiert Teilmenge der Beobachtungen, `na.action=` Aktion bei fehlenden Werten: `"na.fail"`, `"na.omit"`, oder eine Funktion. Die folgenden generischen Funktionen können meistens auf geschätzte Modelle angewendet werden:

predict(fit,...) berechnet Werte von `fit` entsprechend der Inputdaten

df.residual(fit) Freiheitsgrad der Residuen

coef(fit) Schätzungen der Koeffizienten (manchmal mit zugehörigen Standardfehlern)

residuals(fit) Residuen des Modells

deviance(fit) Streuung

fitted(fit) Schätzungen der Abhängigen

logLik(fit) liefert logLikelihood-Wert und die Anzahl der Parameter

AIC(fit) berechnet Akaike-Informationskriterium

Statistik

aov(formula) Varianzanalyse

anova(fit,...) Analyse der Varianzen (oder Streuungen) für ein oder mehrere geschätzte Modelle

density(x) bestimme Dichtefunktion von `x`

binom.test(), **pairwise.t.test()**, **power.t.test()**, **prop.test()**, **t.test()**, ... use `help.search("test")`

Verteilungen

rnorm(n, mean=0, sd=1) Gauß (normal)

rexp(n, rate=1) exponential

rgamma(n, shape, scale=1) Gamma

rpois(n, lambda) Poisson

rweibull(n, shape, scale=1) Weibull

rcauchy(n, location=0, scale=1) Cauchy

rbeta(n, shape1, shape2) Beta

rt(n, df) 'Student' (*t*)

rf(n, df1, df2) Fisher–Snedecor (*F*) (χ^2)

rchisq(n, df) Pearson

rbinom(n, size, prob) binomial

rgeom(n, prob) geometrisch

rhyper(nn, m, n, k) hypergeometrisch

rlogis(n, location=0, scale=1) logistisch

rlnorm(n, meanlog=0, sdlog=1) lognormal

rnbinom(n, size, prob) negative binomial

runif(n, min=0, max=1) uniform

rwilcox(nn, m, n), `rsignrank(nn, n)` Wilcoxon-Statistik

Zu allen diesen Funktionen gehören weitere. Dabei wird der Anfangsbuchstabe `r` ersetzt durch `d`, `p` or `q` für die Berechnung der Wahrscheinlichkeit (Dichte) (`dfunc(x, ...)`), der kumulativen Wahrscheinlichkeit (`pfunc(x, ...)`), und des Quantils (`qfunc(p, ...)`, mit $0 < p < 1$).

Programmierung

function(arglist) expr Funktionsdefinition

return(value)

if(cond) expr

if(cond) cons.expr else alt.expr

for(var in seq) expr

while(cond) expr

repeat expr

break

next

Geschweifte Klammern {} für Anweisungsblöcke

ifelse(test, yes, no) Wert mit gleicher Gestalt wie `test`, gefüllt mit Elementen entweder `yes` oder `no`

do.call(funname, args) Funktionsaufruf mit Hilfe von Funktionsname und der Liste der zu übergebenden Argumente