# R-Referenzkarte

## Hilfe

Zu allen **R**-Funktionen gibt es eine online Dokumentation.

`?Funktionsname` online Dokumentation zur Funktion `Funktionsname`

`help.search("'name"')` Funktion suchen

`help.start()` Führt zu einem web-basiertem Hilfesystem

## Vektoren

`t.v <- c(2,-1,9,0.4,100)` erzeugt den Vektor $[2, -1, 9, 0.4, 100]$ und speichert ihn unter dem Namen `t.v`

`rep(0,10)` Vektor von 10 Nullen

`rep(t.v,3)` Wiederhole `t.v` 3 mal

`t.vs <- seq(0,2,0.5)` Folge von Zahlen von 0 bis 2 mit Inkrement 0.5, also $[0, 0.5, 1, 1.5, 2]$

`t.i <- 1:7` Abkürzung für $[1, 2, \ldots, 7]$ (das wird mit einem eigenen Spezialzeichen abgekürzt, weil es so oft gebraucht wird).

`rep` und `seq` können einiges mehr, siehe `?rep`, `?seq`

## Matrizen

`t.m <- matrix(t.v,5,3)` Matrix der Dimension $5 \times 3$, spaltenweise gefüllt mit dem Inhalt von `t.v`

`t.m <- cbind(1:5,t.v,c(0,0,2,2,3))` spaltenweise zusammenfügen

`rbind` zeilenweise zusammenfügen

## Umwandlung

Man kann einige Typen sinnvoll in einander umwandeln:

- numeric in character (und manchmal umgekehrt)
- logical in numeric (0 für FALSE, 1 für TRUE)

`as.character(1:3)` ergibt "1" "2" "3"

`as.logical(c(0,1,-1,3,0.5))` ergibt FALSE TRUE TRUE TRUE TRUE

## Klassen von Objekten

Diese grundlegenden Typen werden ergänzt durch die Möglichkeit, beliebige weitere Typen zu definieren. Man spricht von Klassen von Objekten. Sie ermöglichen eine bestimmte Art von „objektorientiertem Programmieren". In der neuesten Version von **R** hat jedes Objekt eine Klasse. Auskunft erteilt `class(t.m)`

## Arithmetische Operationen

Die arithmetischen Operatoren + - * / ^ werden auf zwei Vektoren oder Matrizen elementweise angewandt.

`c(2,4)*c(3,-1)` liefert 6 -4

## Recycling

Hat das eine Argument mehr Elemente als das andere, dann werden die Elemente des kürzeren zyklisch wiederholt.

`c(2,4)*3` liefert 6 12

## Uneigentliche Werte

**R** kennt auch die uneigentlichen Werte

`Inf` und `-Inf` : unendlich
`NaN` : unbestimmt
`NA` : fehlend oder undefiniert

## Mathematische Funktionen

Z.B. `log10(t.v)`, wird ebenfalls elementweise ausgeführt. Negative Elemente von `t.v` ergeben `NA`.
Die üblichen Funktionen sind vorhanden: `log`, `exp`, `sin`, `cos`, `tan`, `ctg`, `asin`, ...

`min(t.v, 500, 1:10)` liefert das Minimum über alle Elemente aller Argumente.

`which.min(t.v)` : Index des (ersten) Elements, das gleich dem Minimum ist.

`pmin` elementweises Minimieren von Vektoren.

## Logische Operationen

Die logischen Werte heissen `TRUE` und `FALSE`. Für Schreibfaule sind die Objekte `T` und `F` gespeichert. Die logischen Operatoren heissen

`!` : not
`&` : and
`|` : or

Logische Objekte sind natürlich wichtig für `if(logical)` - Konstruktionen.

## Vergleiche

`==` , `!=` , `>` , `<` , `>=` , `<=`
Aufgepasst mit `t.v<-3`! Das überschreibt `t.v`. Den Vergleich mit -3 erhält man mit einem Abstand: `t.v< -3`.

## Zusammenfassende Funktionen

`sum(t.v)` : Die Summe der Elemente von `t.v`

`mean(t.v)` : Der Durchschnitt der Elemente von `t.v`

`median(t.v)` : Der Median der Elemente von `t.v`

`var(t.v)` : Die Varianz der Elemente von `t.v` (Kalkuliert für $n-1$)

`cov(x,y)` : Die Kovarianz von `x` und `y`

`sd(t.v)` : Standardabweichung von `t.v`

`quantile(t.v)` : Quantile (0%, 25%, 75%, 100%) von `t.v`

## Normalverteilung

`dnorm(7,5,2)` : Dichte der Normalverteilung für $x = 7, \mu = 5, \sigma = 2$.

`pnorm(7,5,2)` : $P(X \leq 7)$ für $X \sim \mathcal{N}(5, 2^2)$.

`qnorm(c(0.025, 0.5, 0.975))` : Quantile der Standard-Normalverteilung.

`rnorm(20)` : Erzeuge 20 standard-normalverteilte Zufallszahlen.

## Andere Verteilungen

Es gibt die entsprechenden Funktionen für alle bekannten Verteilungen.

`d...(x)` : Dichte respektive, bei diskreten Verteilungen, Wahrscheinlichkeit für den Wert $x$.

`p...(x)` : Kumulative Verteilungsfunktion, nützlich zur Berechnung von $p$-Werten: `1-pchisq(3.6,2)`.

`q...(x)` : Quantile (inverse kumulative Verteilungsfunktion), nützlich zur Berechnung von kritischen Werten für Tests: `qchisq(0.95,2)`

`r...(n)` : $n$ Zufallszahlen gemäss der Verteilung (siehe Zufallszahlen).

Es gibt folgende **Verteilungen**: `beta`, `binom`, `cauchy`, `chisq`, `exp`, `f`, `gamma`, `geom`, `hyper`, `lnorm`, `logis`, `nbinom`, `norm`, `pois`, `t`, `unif`, `weibull`, `wilcox` und weitere in verschiedenen speziellen packages.

## Zufallszahlen

`runif(5)` : erzeugt 5 Zufallszahlen gemäss der uniformen Verteilung in (0,1).

`rnorm(20, mean=5, sd=2)` : 20 normalverteilte Zufallszahlen mit Erwartungswert 5 und Standardabweichung 2.

`set.seed(28)` : Startwert für (Pseudo-) Zufallszahlen-Generator festlegen. Führt zu reproduzierbaren Ergebnissen.

`sample(8)` : Zufällige Reihenfolge der Zahlen 1 bis 8.

## Daten einlesen

Um Datensätze mit **R** zu analysieren, müssen diese zuerst eingelesen werden. Je nach Struktur des Datenfiles kommen verschiedene Einlese-Befehle zur Anwendung. Es können u.a. Textfiles (.txt, .dat), Datenfiles (.csv) und Excel-Files eingelesen werden.

```
read.csv("C:/Daten.csv", header=T)
```

```
read.csv("C:\\Daten.csv", header=T)
```

```
read.table("C:/My Documents/Daten.txt")
```

**Wichtige Argumente:**

`header=TRUE/FALSE`: Falls die Bezeichnungen der Variablen in der ersten Datenzeile stehen, wählen Sie `header=TRUE`, ansonsten `header=FALSE`

`sep=` : Das Trennzeichen zwischen den Variablen (Datenfile in einem Editor anschauen!) muss mit dem Argument `sep="..."` angegeben werden. Das Trennzeichen „Tabulator“ wählen Sie mit `sep="\t"`

`dec=` : Mit `dec=","` kann das Dezimaltrennzeichen eingegeben werden, falls es nicht „.“ ist

`fill=TRUE`: Beim Abspeichern von Excel-Tabellen als .csv-Files kommt es vor, dass einzelne Zeilen am Ende ein (leeres) Feld zu wenig haben, was zu einer Fehlermeldung beim Einlesen führt. Dieser Fehler wird mit dem Argument `fill=TRUE` verhindert

## Grafiken

**Eindimensional**:

`hist`, `barplot` : Histogramm und Stabdiagramm

`qqnorm`, `qqplot` : QQ-Diagramme zur Beurteilung von Verteilungen

**Zweidimensional**:

`plot` : Streudiagramm

**Wichtigste allgemeine Elemente der Grafik-Funktionen:**

`type="p"`: Es sollen Punkte gezeichnet werden, `type="l"`: Linien, `type="b"`: beides

`xlim=`, `ylim=` : Begrenzung der Koordinaten. Sie werden nur exakt so verwendet, wenn man auch `xaxs="i"`, `yaxs="i"` setzt

`xlab="x"`, `ylab="y"` : Achsenbezeichnung

`main="Titel"` : Titel

**Wichtige grafische „low level“-Funktionen:**
Typischerweise schaltet man Teile der high level grafischen Funktion ab, indem man schreibt: `plot(..., type="n", axes=FALSE)` (keine Punkte resp. keine Achsen zeichnen). Dann ruft man die entsprechenden Funktionen auf:

`points`: Punkte zeichnen

`lines`, `arrows`, `segments`: Linien zeichnen

`text`: Punkte mit Text beschriften

`axis`: Achsenbeschriftung

`box`: Umrandung

`mtext`, `title`: Text im Rand der Figur

`rug`: Daten am Rand zeichnen durch Striche

`abline`: Gerade zeichnen

`polygon`: Polygon zeichnen und einfärben oder schraffieren

`legend`: Legende

**Wichtigste grafische Parameter:**

`lty`, `lwd` : Linientyp und -dicke

`col`: Farbe

**Mehrere Diagramme (frames) auf einer Seite:**

`par(mfrow=c(2,3))` : Aufteilung in 2 Zeilen und 3 Spalten, zeilenweise gefüllt.

`par(mfcol=c(2,3))` : Aufteilung in 2 Zeilen und 3 Spalten, spaltenweise gefüllt