Grundlagen

Jan-Philipp Kolb

Friday, March 20, 2015

R ist eine Objekt-orientierte Sprache

Vektoren und Zuweisungen

- R ist eine Objekt-orientierte Sprache

```
# <- ist der Zuweisungsoperator

b <- c(1,2)
```

- erzeugt ein Objekt mit den Zahlen 1 und 2
- Eine Funktion kann auf dieses Objekt angewendet werden:

```
mean(b)
```

```
## [1] 1.5
```

- berechnet den Mittelwert

Mit den folgenden Funktionen können wir etwas über die Eigenschaften des Objekts lernen:

```
length(b) # b hat die Länge 2
```

```
## [1] 2
```

```
str(b) # b ist ein numerischer Vektor
```

```
##  num [1:2] 1 2
```

Funktionen im base-Paket

Diese Funktionen brauchen nur ein Argument.

```
# Länge
b <- c(1,2)
length(b)
```

```
## [1] 2
```

```
  # Das Maximum:
max(b)
```

```
## [1] 2
```

```r
  # Minimum
min(b)
```

```
## [1] 1
```

```r
  # Standardabweichung
sd(b)
```

```
## [1] 0.7071068
```

```r
  # Varianz
var(b)
```

```
## [1] 0.5
```

```r
  # Mittelwert
mean(b)
```

```
## [1] 1.5
```

```r
  # Median
median(b)
```

```
## [1] 1.5
```

```r
  # Das Ergebnis kann wieder einem Objekt zugewiesen werden
med_b <- median(b)
```

Funktionen mit mehr Argumenten:

```r
d <- c(1,4,3,7,9,5,4,3)


  # Quantil berechnen:
quantile(d,0.9)
```

```
## 90%
## 7.6
```

```r
  # hier werden 2 Elemente aus d gezogen
sample(x=d,size=2,replace=FALSE)
```

```
## [1] 5 9
```

```r
  # jedes Mal können andere Ergebnisse resultieren

  # hier wird nur ein Element gezogen
sample(x=d,size=1,replace=FALSE)
```

```
## [1] 7
```

```
  # auch dieses Ergebnis kann wieder in einem Element
  # gespeichert werden

stichA <- sample(x=d,size=1,replace=FALSE)
```

Das Argument replace gibt an, ob eine Stichprobe mit oder ohne zurücklegen gezogen wird

Eine einführende Übersicht findet man unter:

http://cran.r-project.org/doc/manuals/R-intro.html

Verschiedene Datentypen

```
A <- c(5,4,3)
is.numeric(A)
```

```
## [1] TRUE
```

```
str(A)
```

```
##  num [1:3] 5 4 3
```

```
B <- c(T,F,T,T)
is.logical(B)
```

```
## [1] TRUE
```

```
str(B)
```

```
##  logi [1:4] TRUE FALSE TRUE TRUE
```

```
C <- c("AB","F","23")
is.logical(C)
```

```
## [1] FALSE
```

```
str(C)
```

```
##  chr [1:3] "AB" "F" "23"
```

```
  # immer das niedrigste
  # Niveau wird genommen

D <- c(1,3,"A")
str(D)
```

```
##  chr [1:3] "1" "3" "A"
```

```r
b <- c(1,2)
log <- c(T,F)
char <- c("A","b")

  # Faktoren sind eine spezielle Form,
  # vor allem bei Regression hilfreich

fac <- as.factor(c(1,2))

  # mit as... kann man also umwandeln

as.character(b)
```

```
## [1] "1" "2"
```

```r
  # wenn man das nicht in Objekt speichert
  # merkt es es sich R auch nicht

b
```

```
## [1] 1 2
```

```r
b <- c(1,2) # numeric
log <- c(T,F) # logical
char <-c("A","b") # character
fac <- as.factor(c(1,2)) # factor
```

Indizieren

```r
####
# vector

A1 <- c(1,2,3,4)
A1
```

```
## [1] 1 2 3 4
```

```r
A1[1]
```

```
## [1] 1
```

```r
A1[4]
```

```
## [1] 4
```

```r
A1[1:3]
```

```
## [1] 1 2 3
```

```r
A1[-4]
```

```
## [1] 1 2 3
```

```r
####
# dataframe

  # Beispieldaten generieren:
AGE <- c(20,35,48,12)
SEX <- c("m","w","w","m")

  # Diese beiden Vektoren zu einem data.frame verbinden:
Daten <- data.frame(Alter=AGE,Geschlecht=SEX)

  # Anzahl der Zeilen/Spalten herausfinden

nrow(Daten) # Zeilen
```

```
## [1] 4
```

```r
ncol(Daten) # Spalten
```

```
## [1] 2
```

```r
AA <- 4:1
A2 <- cbind(A1,AA)
A2[1,1]
```

```
## A1
##  1
```

```r
A2[2,]
```

```
## A1 AA
##  2  3
```

```r
A2[,1]
```

```
## [1] 1 2 3 4
```

```r
A2[,1:2]
```

```
##      A1 AA
## [1,]  1  4
## [2,]  2  3
## [3,]  3  2
## [4,]  4  1
```

Matrizen und Arrays

- In Matrizen und Arrays stehen meist nur numerische Werte.
- Dadurch wird beispielsweise Matrix Multiplikation möglich.
- Anders als beim data.frame sind mehr als zwei Dimensionen möglich.

```
A <- matrix(seq(1,100), nrow = 4)
dim(A)
```

```
## [1]  4 25
```

```
A3 <- array(1:8,c(2,2,2))
A3
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```
A3[,,2]
```

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

Liste

```
    * Eine Liste in R entspricht einem geschachtelten Array in anderen Programmiersprachen
        * Listen können alles enthalten
        * Listen können geschachtelt sein
        * Listen sollte man sehr bedacht verwenden
```

```
A4 <- list(A1,1)
A4
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 1
```

```r
A4[[2]]
```

```
## [1] 1
```

Sequenzen

```r
# Sequenz von 1 bis 10
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
seq(-2,8,by=1.5)
```

```
## [1] -2.0 -0.5  1.0  2.5  4.0  5.5  7.0
```

```r
a<-seq(3,12,length=12)

b<- seq(to=5,length=12,by=0.2)

d <-1:10
d<- seq(1,10,1)
d <- seq(length=10,from=1,by=1)

# wiederhole 1 10 mal
rep(1,10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```r
rep("A",10)
```

```
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Die Funktion paste

```r
?paste
```

```
## starting httpd help server ... done
```

```r
paste(1:4)
```

```
## [1] "1" "2" "3" "4"
```

```r
paste("A", 1:6, sep = "")
```

```
## [1] "A1" "A2" "A3" "A4" "A5" "A6"
```

- Bullet 1
- Bullet 2
- Bullet 3