Statistische Software (R)

Paul Fink, M.Sc.

Institut für Statistik
Ludwig-Maximilians-Universität München

Vektoren, Matrizen

Grundlegende Operatoren und Funktionen

Aufruf der Hilfeseiten zu grundlegende Operatoren und Funktionen:

`?Arithmetic`	Grundlegende Operatoren für numerische Vektoren
`?Logic`	Operatoren für logische Vektoren
`?log`	Logarithmus und Exponens
`?Trig`	Trigonnometrische Funktionen
`?Special`	Z.B. Binomialkoeffizienten, Fakultät, etc.

Konstanten

Übersicht einiger Konstanten:

`pi`	Die Zahl π
`Inf`, `-Inf`	$\infty, -\infty$
`NaN`	Not a Number: z.B. $0/0$
`NA`	Not available: fehlende Werte
`NULL`	leere Menge
`letters`	Kleinbuchstaben von a bis z
`LETTERS`	Großbuchstaben von A bis Z

Datentypen in R

DAS Datenobjekt ist ein Vektor mit Elementen des Typs

- **numeric**: ganzzahlige oder Gleitkomma-Werte,
- **character**: beliebige Zeichen,
- **logical**: die Zustände `TRUE` und `FALSE`,
- **list**: ein Objekt beliebigen Typs, auch wieder eine Liste (rekursive Datenstruktur!). Mehr dazu später.

Jeder Vektor besitzt Elemente **eines!!** Typs und hat eine Länge (`length()`).

Vektoren unterschiedlichem Typs

- Vektor vom Typ **numeric**.

```
> numvec <- c(2.54, 4.22, 2.99, 3.14, 3.44)
> numvec
[1] 2.54 4.22 2.99 3.14 3.44
```

- Vektor vom Typ **character**.

```
> charvec <- c("Statistische", "Software")
> charvec
[1] "Statistische" "Software"
```

- Vektor vom Typ **logical**.

```
> logicvec <- c(TRUE, FALSE, FALSE, TRUE)
> logicvec
[1]  TRUE FALSE FALSE  TRUE
```


Automatische Umwandlung

- Konstruktion einfacher Vektoren

```
> c(1, 2, 7)
> c("Hallo", "Welt")
```

- R wandelt den Typ eines Objektes *automatisch* um, wenn dies notwendig und möglich ist:

```
> TRUE + 2
[1] 3
> c("Hello", sqrt(3))
[1] "Hello"            "1.73205080756888"
> c(1:2, 3.14, exp(1i * pi))
[1]  1.00+0i  2.00+0i  3.14+0i -1.00+0i
```


Automatische Umwandlung

- Beispiele für häufige Umwandlungen sind

logisch	$\Longrightarrow$	numerisch
logisch, numerisch	$\Longrightarrow$	Text
numerisch	$\Longrightarrow$	logisch

```
> as.numeric(rnorm(10) >= 1)
 [1] 1 0 0 0 0 0 0 0 0 0
> as.logical(c(0, pi))
[1] FALSE  TRUE
> c(2, "Hallo", TRUE)
[1] "2"     "Hallo" "TRUE"
```


Rechnen mit Vektoren

- Wie in Linearer Algebra komponentenweise Addition und Subtraktion

```
> x <- 1:4
> y <- c(4,10,2,0)
> x + y
[1]  5 12  5  4
```

- Achtung: Multiplikation/Division auch komponentenweise!!

```
> x * y
[1]  4 20  6  0
```

- **Wichtig:** Die meisten Operationen von 2 Vektoren werden komponentenweise durchgeführt!!

Recycling-Regel

R erlaubt auch Rechnen mit Vektoren unterschiedlicher Länge.

```
> x
[1] 1 2 3 4
> x + c(1, 2)
[1] 2 4 4 6
```

entspricht

```
> x + c(1, 2, 1, 2)
[1] 2 4 4 6
```

Fehlende Werte werden aus bestehenden „recycled“.

Recycling-Regel

Funktioniert auch wenn Vektorlängen nicht Vielfache sind, allerdings mit Warnung

```
> x + c(1, 2, 4) # x + c(1, 2, 4, 1)
Warning in x + c(1, 2, 4): Länge des längeren Objektes
ist kein Vielfaches der Länge des kürzeren Objektes
[1] 2 4 7 5
```


Zugriff auf Vektorelemente

1. Vektor von positiven Zahlen

```
> letters[1:3]
[1] "a" "b" "c"
> letters[c(2, 4, 6)]
[1] "b" "d" "f"
```

2. Logischer Vektor

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x[(x > 5)]
[1]  6  7  8  9 10
> x[((x %% 2) == 0)]
[1]  2  4  6  8 10
```


Zugriff auf Vektorelemente

3. Vektor von negativen Zahlen

```
> x <- 1:10
> x[-(1:5)]
[1]  6  7  8  9 10
```

4. Vektor von Zeichenketten

   Die Elemente eines Vektors kann man mit Namen versehen. Mittels dieses Namens kann auf die Elemente zugegriffen werden.

```
> x <- c(Wasser = 1, Saft = 2, Limonade = 3 )
> names(x)
[1] "Wasser"   "Saft"     "Limonade"
> x["Saft"]
Saft
   2
```

Zugriff auf Vektorelemente

5. Leerer Index. Diesen haben wir bereits ständig verwendet!

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x[]
 [1]  1  2  3  4  5  6  7  8  9 10
```


Faktoren

- Nominale oder ordinale Merkmale werden in R als „Faktoren“ codiert

```
> x <- factor(c("Saft", "Saft", "Limonade", "Saft", "Wasser"))
> x
[1] Saft     Saft     Limonade Saft     Wasser
Levels: Limonade Saft Wasser
```

- Den einzelnen Stufen werden ganzzahlige Werte zugeordnet, wie der `str` Befehl zeigt:

```
> str(x)
 Factor w/ 3 levels "Limonade","Saft",..: 2 2 1 2 3
```


Faktoren

- Bestimmte Reihenfolge der Levels festlegen mit Argument `levels`:

```
> x <- factor(c("Saft", "Saft", "Limonade", "Saft", "Wasser"),
+             levels = c("Saft", "Wasser", "Limonade"))
> x
[1] Saft     Saft     Limonade Saft     Wasser
Levels: Saft Wasser Limonade
> str(x)
 Factor w/ 3 levels "Saft","Wasser",..: 1 1 3 1 2
> levels(x)
[1] "Saft"     "Wasser"   "Limonade"
```

- Nachträgliches Ändern des ersten Levels

```
> x <- relevel(x, "Wasser")
> str(x)
 Factor w/ 3 levels "Wasser","Saft",..: 2 2 3 2 1
```


Faktoren

- Ein Vektor von Typ `character` kann in einen Faktor mittels `as.factor()` umgewandelt werden:

```
> x <- c("Apfel", "Birne", "Apfel", "Traube", "Traube", "Kiwi")
> x <- as.factor(x)
> x
[1] Apfel  Birne  Apfel  Traube Traube Kiwi
Levels: Apfel Birne Kiwi Traube
```

Sequenzen

Der Befehl `seq()`

- Absteigende Sequenz mit gleicher Schrittweite:

```
> seq(from = 3, to = -2, by = -0.5)
 [1]  3.0  2.5  2.0  1.5  1.0  0.5  0.0 -0.5 -1.0 -1.5 -2.0
```

- Standardschrittweite ist $+1$ oder -1:

```
> seq(from = 2, to = 4)
[1] 2 3 4
> seq(from = 4, to = 2)
[1] 4 3 2
> 2:4
[1] 2 3 4
```


Sequenzen

- Sequenzen mit vorgegebener Länge:

```
> seq(to = 10, length = 10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(from = 10, length = 10)
 [1] 10 11 12 13 14 15 16 17 18 19
> seq(from = 10, length = 10, by = 0.1)
 [1] 10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
```


Wiederholungen

Der Befehl `rep()`

- n-malige Wiederholung eines Objekts:

```
> rep(3.5, times = 10)
 [1] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
> rep(1:4, times = 2)
[1] 1 2 3 4 1 2 3 4
> (anz <- seq(from = 2, to = 8, by = 2))
[1] 2 4 6 8
> rep(1:4, times = anz)
 [1] 1 1 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 4 4
> rep(3.5, 10)
 [1] 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
```


Wiederholungen

- Jedes Vektorelement wird mehrmals hintereinander wiederholt:

```
> rep(1:4, each = 2)
[1] 1 1 2 2 3 3 4 4
> rep(1:4, each = 2, times = 3)
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

Aufgaben

1.) Erstellen Sie einen Vektor `ung`, welcher die ersten 15 ungeraden Zahlen enthält die echt größer als 107 sind!

2.) Geben Sie vom Vektor `ung` diejenigen Zahlen aus, die durch 3 teilbar sind und bilden sie die Summe daraus!

3.) Erstellen Sie einen Faktor `faecher`, der 24 mal das Wort `Statistik`, 1 mal `Informatik` und 5 mal `Mathematik` enthält. Statistik, soll dabei an erster Stelle der Faktorlevels stehen.

Matrizen

Eine Matrix in R ist ein Vektor mit Dimensions-Attribut!!!

Erzeugen einer Matrix:

```
> x <- matrix(nrow = 4, ncol = 2, byrow = TRUE,
+             data = c(1, 2, 3, 4, 5, 6, 7, 8))
> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```

Auf ein einzelnes Element kann mittels der Notation `x[i,j]` zugegriffen werden (*i* ist die Zeile und *j* ist die Spalte):

```
> x[3, 2]
[1] 6
```


Matrizen

Was machen die Argumente in der oben angegebenen Funktion?

```
?matrix
```


Matrizen

Wir können nun bestimmte *Eigenschaften* der Matrix abfragen:

```
> dim(x)
[1] 4 2
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

D.h., neben den Elementen selbst werden zusätzliche, abfragbare Eigenschaften im *Objekt* Matrix abgelegt (Dimension, Anzahl der Zeilen, Anzahl der Spalten).

Spaltenweise Vektoren und Matrizen verbinden mit `cbind()`

```
> y <- c(12, 3, 4, 1)
> cbind(x, y)
          y
[1,] 1 2 12
[2,] 3 4  3
[3,] 5 6  4
[4,] 7 8  1
> cbind(y, y)
      y  y
[1,] 12 12
[2,]  3  3
[3,]  4  4
[4,]  1  1
```


Zeilenweise Vektoren und Matrizen verbinden mit `rbind()`

```
> rbind(c(100, 0), x)
     [,1] [,2]
[1,]  100    0
[2,]    1    2
[3,]    3    4
[4,]    5    6
[5,]    7    8
> rbind(x, x)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    1    2
[6,]    3    4
[7,]    5    6
[8,]    7    8
```