Computergestützte Datenanalyse

BSc Psychologie SoSe 2021

Prof. Dr. Dirk Ostwald

(3) Vektoren, Matrizen, Arrays, Faktoren

Literaturhinweise

Die Diskussion folgt Cotton (2013, Kapitel 4) und Wickham (2019, Kapitel 3).

Vektoren, Matrizen, Arrays, Faktoren

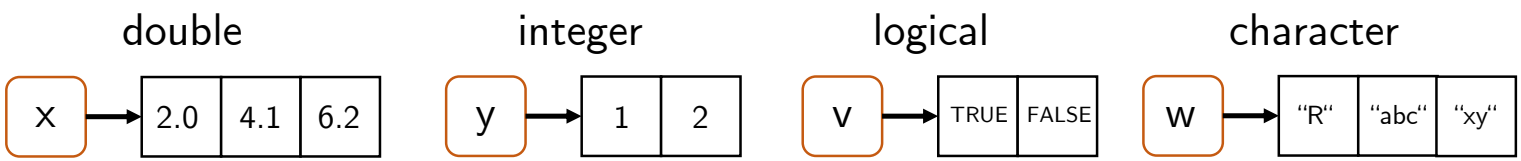
- Vektoren
- Matrizen
- Arrays
- Faktoren
- Übungen und Selbstkontrollfragen

Vektoren, Matrizen, Arrays, Faktoren

- **Vektoren**
- Matrizen
- Arrays
- Faktoren
- Übungen und Selbstkontrollfragen

Übersicht

- Vektoren sind geordnete Folgen von Werten.
- Die einzelnen Werte eines Vektors heißen Elemente.
- Vektoren, deren Elemente vom gleichen Typ sind, heißen **atomic vectors**.
- Die zentralen Typen sind **numeric** (**double**, **integer**), **logical**, **character**.



- Mit dem Begriff **Vektor** ist hier **atomic vector** gemeint.

Elementarwerte

Numeric

- Double wird in Dezimalnotation oder wissenschaftlicher Notation spezifiziert
- Weitere mögliche Werte sind Inf, -Inf, und NaN (Not-a-Number)

```
x = 1                    # Einelementiger Vektor vom Typ double
y = 2.1e2                # Einelementiger Vektor vom Typ double
z = Inf                  # Einelementiger Vektor vom Typ double
```

- Integer wird wie double ohne Dezimalstellen spezifiziert, gefolgt von L (long integer)

```
x = 1L                   # Einelementiger Vektor vom Typ integer
y = 200L                 # Einelementiger Vektor vom Typ integer
```

Logical

- TRUE oder FALSE, abgekürzt T oder F

```
x = TRUE                 # Einelementiger Vektor vom Typ logical
y = F                    # Einelementiger Vektor vom Typ logical
```

Character

- Anführungszeichen ("a") oder Hochkommata ('a')

```
x = "a"                  # Einelementiger Vektor vom Typ character
y = 'test'               # Einelementiger Vektor vom Typ character
```

Erzeugung

Direkte Konkatenation von Elementarwerten mit **c()**

```
x = c(1,2,3)                # numeric vector [1,2,3]
y = c(0,x,4)                # numeric vector [0,1,2,3,4]
s = c("a", "b", "c")        # character vector ["a", "b", "c"]
l = c(TRUE, FALSE)          # logical  vector [TRUE, FALSE]
```

c() konkateniert die Eingabeargumente und erzwingt einen einheitlichen Datentyp

```
x = c(1, "a", TRUE)         # character vector ["1", "a", "TRUE"]
```

Erzeugen "leerer" Vektoren mit **vector()**

```
v = vector("double",3)      # double vector [0,0,0]
w = vector("integer",3)     # integer vector [0,0,0]
l = vector("logical",2)     # logical vector [FALSE, FALSE]
s = vector("character",4)   # character vector ["", "", "", ""]
```

Erzeugen "leerer" Vektoren mit double(), integer(), logical(), character()

```
v = double(3)               # double vector [0,0,0]
w = integer(3)              # integer vector [0,0,0]
l = logical(2)              # logical vector [FALSE, FALSE]
s = character(4)            # character vector ["", "", "", ""]
```

Erzeugung

Erzeugen von ganzzahligen Sequenzen mithilfe des **Colonoperators**

- **a:b** erzeugt ganzzahlige Sequenzen von a (inklusive) bis b (maximal)

```
x = 0:5                    # [0,1,2,3,4,5]
y = 1.5:6.1                # [1.5, 2.5, 3.5, 4.5, 5.5]
```

Erzeugen von Sequenzen mit **seq()**

- **seq**(from, to, by = ((to - from)/(len - 1), len = NULL, ...)

```
x_1 = seq(0,5)             # wie 0:5, [0,1,2,3,4,5]
x_2 = seq(0,1,len = 5)     # 5 Zahlen zwischen 0 (inkl.) und 1 (inkl.)
                           # [0.00, 0.25, 0.50, 0.75, 1.00]
x_3 = seq(0,2,by = .15)    # 0.15 Schritte zwischen 0 (inkl.) und 2 (max.)
                           # [0.00, 0.15, 0.30, ..., 1.50 1.65 1.80 1.95]
x_4 = seq(1,0,by = -.1)    # -0.1 Schritte zwischen 1 (inkl.) und 0 (min.)
```

- **seq.int()**, **seq_len()**, **seq_along()** als weitere Varianten

```
x_1 = seq.int(0,5)         # wie 0:5, [0,1,2,3,4,5]
x_2 = seq_len(5)           # Natuerliche Zahlen bis 5, [1,2,3,4,5]
x_3 = seq_along(c("a","b")) # wie seq_len(length(c("a", "b")))
```

Charakterisierung

length() gibt die Anzahl der Elemente eines Vektors aus

```
x = 0:10                    # Vektor
length(x)                   # Die Anzahl der Elemente des Vektors
[1] 11                      # ... ist 11
```

typeof() gibt den elementaren Datentyp eines Vektors aus

```
x = 1:3L                    # Vektor
typeof(x)                   # Der Typ des atomic vectors
[1] "integer"               # ... ist integer
y = c(T,F,T)                # Vektor
typeof(y)                   # Der Typ des atomic vectors
[1] "logical"               # ... ist logical
```

- mode(), storage.mode() werden nicht empfohlen, sie existieren für S Kompatibilität

is.logical(), **is.double()**, **is.integer()**, **is.character()** testen den Datentyp

```
is.double(x)                # Testen obigen Vektors
[1] FALSE                   # Der Vektor ist nicht vom Typ double
is.logical(y)               # Testen obigen Vektors
[1] TRUE                    # Der Vektor ist vom Typ logical
```

- is.vector(), is.atomic(), is.numeric() haben spezifischere Funktionen

Typangleichung

Bei Kombination verschiedene Typen wird einheitlicher Datentyp erzwungen (coercion)

Es gilt logical $<$ integer $<$ double $<$ character

```
x = c(1.2,"a")              # Kombination gemischter Elementarwerttypen
[1] "1.2" "a"               # character schlaegt double
typeof(x)                   # Erzeugter Vektor
[1] "character"             # ... ist vom Typ character

y = c(1L, TRUE)             # Kombination  gemischter Elementarwerttypen
[1] 1 1                     # integer schlaegt logical
typeof(y)                   # Erzeugter Vektor
[1] "integer"               # ... ist vom Typ integer
```

as.logical(), **as.integer()**, **as.double()**, **as.character()** erlauben Typangleichung

```
x = c(0,1,1,0)              # double Vektor
y = as.logical(x)           # ... umgewandelt in logical
[1] FALSE  TRUE  TRUE FALSE # logical vector
```

Typangleichung geschieht oft implizit

```
x = c(T, F, T, T)           # logical Vektor
s = sum(x)                  # Summation in integer gewandelter logical Elemente
[1] 3                       # 1 + 0 + 1 + 1
```

Indizierung

Indizierung

- Einzelne oder mehrere Vektorkomponenten werden durch Indizierung adressiert.
- Indizierung wird auch Indexing, Subsetting, oder Slicing genannt.
- Zur Indizierung werden eckige Klammern [ ] benutzt.
- Indizierung kann zur Kopie oder Manipulation von Komponenten benutzt werden.
- Der Index des ersten Elements ist 1 (nicht 0, wie in anderen Sprachen).

```
x     = c("a", "b", "c")    # character vector ["a", "b", "c"]
y     = x[2]                # Kopie von "b" in y
x[3]  = "d"                 # Aenderung von x zu x = ["a", "b", "d"]
```

Prinzipien der Indizierung in R

Indizierung mit ...

- ... einem Vektor positiver Zahlen adressiert entsprechende Komponenten.
- ... mit einem Vektor negativer Zahlen adressiert komplementäre Komponenten.
- ... einem logischen Vektor adressiert die Komponenten mit TRUE.
- ... einem character Vektor adressiert benannte Komponenten.

Indizierung

Beispiele

Indizierung mit einem Vektor positiver Zahlen

```
x = c(1,4,9,16,25)          # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[1:3]                  # 1:3 erzeugt Vektor [1,2,3], x[1:3] = [1,4,9]
z = x[c(1,3,5)]             # c(1,3,5) erzeugt Vektor [1,3,5], x[c(1,3,5)] = [1,9,25]
```

Indizierung mit einem Vektor negativer Zahlen

```
x = c(1,4,9,16,25)          # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[c(-2,-4)]             # Alle Komponenten ausseer 2 und 4, x[c(-2,-4)] = [1,9,25]
z = x[c(-1,2)]              # Gemischte Indizierung nicht erlaubt (Fehlermeldung)
```

Indizierung mit einem logischen Vektor

```
x = c(1,4,9,16,25)          # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[c(T,T,F,F,T)]         # TRUE Komponenten,  x[c(T,T,F,F,T)] = [1,4,25]
z = x[x > 5]                # x > 5 = [F,F,T,T,T], x[x > 5] = [9,16,25]
```

Indizierung mit einem character Vektor

```
x = c(1,4,9,16,25)          # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
names(x) = c("a","b")       # Benennung der Komponenten als [a  b <NA> <NA> <NA>]
y = x["a"]                  # x["a"] = 1
```

Indizierung

R hat eine (zu) hohe Flexibilität bei Indizierung

Out-of-range Indizes verursachen keine Fehler, sondern geben NA aus

```
x = c(1,4,9,16,25)          # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[10]                   # x[10] = NA (Not Applicable)
```

Nichtganzzahlige Indizes verursachen keine Fehler, sondern werden abgerundet

```
y = x[4.9]                  # x[4.9]  = x[4]  = 16
z = x[-4.9]                 # x[-4.9] = x[-4] = [1,4,9,25]
```

Leere Indizes indizieren den gesamten Vektor

```
y = x[]                     # y = x
```

Arithmetik

Unitäre arithmetische Operatoren und Funktionen werden elementweise ausgewertet

```
a    = seq(0,1,len=11)      # a = [ 0.0    ,  0.1    , ...,  0.9     , 1.0    ]
b    = -a                   # b = [-0.0    , -0.1    , ..., -0.9     ,-1.0    ]
v    = a^2                  # v = [ 0.0^2  ,  0.1^2  , ...,  0.9^2   , 1.0^2  ]
w    = log(a)               # w = [ln(0.0), ln(0.1), ..., ln(0.9), ln(1.0)]
```

Binäre arithmetische Operatoren werden elementweise ausgewertet

Vektoren gleicher Länge

```
a = c(1,2,3)                # a = [1,2,3]
b = c(2,1,4)                # b = [2,1,4]
v = a + b                   # v = [1,2,3] + [2,1,4] = [1+2,2+1,3+4] = [   3,  3,    7]
w = a - b                   # w = [1,2,3] - [2,1,4] = [1-2,2-1,3-4] = [  -1,  1,   -1]
x = a * b                   # x = [1,2,3] * [2,1,4] = [1*2,2*1,3*4] = [   2,  2,   12]
y = a / b                   # y = [1,2,3] / [2,1,4] = [1/2,2/1,3/4] = [0.50,  2, 0.75]
```

Vektoren und Skalare (= Vektoren der Länge 1)

```
a = c(1,2,3)                # a = [1,2,3]
b = 2                       # b = [2]
v = a + b                   # v = [1,2,3] + [2,2,2] = [1+2,2+2,3+2] = [   3,  4,    5]
w = a - b                   # w = [1,2,3] - [2,2,2] = [1-2,2-2,3-2] = [  -1,  2,    1]
x = a * b                   # x = [1,2,3] * [2,2,2] = [1*2,2*2,3*2] = [   2,  4,    6]
y = a / b                   # y = [1,2,3] / [2,2,2] = [1/2,2/2,3/2] = [0.5,  1,  1.5]
```

Recycling

R erlaubt (leider) auch Arithmetik mit Vektoren unterschiedlicher Länge

Bei ganzzahligen Vielfachen der Länge wird der kürzere Vektor wiederholt

```
x = 1:2                    # x = [1,2], length(x) = 2
y = 3:6                    # y = [3,4,5,6], length(y) = 4
v = x + y                  # v = [1,2,1,2] + [3,4,5,6] = [4,6,6,8]
```

Arithmetik von Vektoren und Skalaren ist ein Spezialfall dieses Prinzips

Andernfalls werden die ersten Komponenten des kürzeren Vektors wiederholt

```
x = c(1,3,5)               # x = [1,3,5], length(x) = 3
y = c(2,4,6,8,10)          # y = [2,4,6,8,10], length(y) = 5
v = x + y                  # v = [1,3,5,1,3] + [2,4,6,8,10] = [3,7,11,9,13]
```

Generell sollten nur Vektoren gleicher Länge arithmetisch verknüpft werden!

Fehlende Werte (NA)

Fehlende Werte werden in R mit NA (not applicable) repräsentiert

Das Rechnen mit NAs ergibt (meist) wieder NA

```
3 * NA                    # Multiplikation eines NA Wertes
[1] NA                    # ... ergibt NA
NA < 2                    # Relationaler Vergleich eines NA Wertes
[1] NA                    # ... ergibt NA

NA^0                      # NA hoch 0
[1] 1                     # ... ergibt 1, weil jeder Wert hoch 0 eins ergibt (?)
NA & FALSE                # NA UND FALSE
[1] FALSE                 # ... ergibt FALSE, weil jeder Wert UND FALSE FALSE ergibt
```

Auf NA testet man mit is.na()

```
x = c(NA, 5, NA, 10)      # Vektor mit NAs
x == NA                   # Kein Testen auf NAs
[1] NA NA NA NA           # 5 == NA ist NA, nicht FALSE

is.na(x)                  # Logisches Testen auf NA
[1] TRUE FALSE TRUE FALSE # Resultat
```

Attribute

Attribute sind Metadaten von R Objekten in Form von Schlüssel-Wert-Paaren

attributes() ruft alle Attribute eines Objektes auf

Perse haben atomic vectors keine Attribute

```
a = 1:3                 # ein numerischer Vektor
attributes(a)           # Aufrufen aller Attribute
NULL                    # perse hat ein atomic vector keine Attribute
```

attr() kann zum Aufrufen und Definieren von Attributen genutzt werden

```
attr(a, "S") = "W"      # a bekommt Attribut mit Schluessel S und Wert W
attr(a, "S")            # Das Attribut mit Schluessel S
[1] "W"                 # ... hat den Wert W
```

Attribute werden bei Operationen oft entfernt (Ausnahmen sind **names** und **dim**)

```
b = a[1]                # Kopie des ersten Elements von a in Vektor b
attributes(b)           # Aufrufen aller Attribute von b
NULL                    # Das Attribut S mit Wert W wurde nicht mit kopiert
```

Attribute

Spezifikation des Attributs **names** gibt den Elementen eines Vektors Namen

```
v = c(x=1,y=2,z=3)          # Elementnamengeneration bei Vektorerzeugung
x y z                       # Elementnamenzeile
1 2 3                       # Elementwertezeile
```

Die Namen können zur Indizierung benutzt werden

```
v["y"]                      # Indizierung per Namen
y                           # Elementname
2                           # Elementwert
```

names() kann zum Definieren und Aufrufen von Namen benutzt werden

```
y = 4:6                     # Erzeugung eines Vektors
names(y) = c("a","b","c")
names(y)                    # Elementnamenaufruf
[1] "a" "b" "c"             # Elementnamen
```

Benannte Namen können hilfreich sein, wenn der Vektor eine Sinneinheit bildet

```
p = c(age = 31, height = 198, weight = 75)
age height weight           # Alter (Jahre), Groesse (cm), Gewicht (kg) einer Person
 31    198     75
```

Vektoren, Matrizen, Arrays, Faktoren

- Vektoren
- **Matrizen**
- Arrays
- Faktoren
- Übungen und Selbstkontrollfragen

Übersicht

- Matrizen sind zweidimensionale, rechteckige Datenstrukturen der Form

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1n_c} \\ m_{21} & m_{22} & \cdots & m_{2n_c} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n_r1} & m_{n_r2} & \cdots & m_{n_rn_c} \end{pmatrix} \tag{1}$$

- Die Elemente $m_{ij}, i = 1, ..., r, j = 1, ..., c$ sind vom gleichen Typ.
- n_r ist die Anzahl der Zeilen (rows), n_c ist die Anzahl der Spalten (columns).
- Jedes Element einer Matrix hat einen Zeilenindex i und einen Spaltenindex j.
- Intuitiv sind Matrizen numerisch indizierte Tabellen.
- Formal sind Matrizen in R zweidimensional interpretierte atomic vectors.
- Matrizen in R sind nicht identisch mit dem mathematischen Matrixbegriff.
- Matrizen in R können allerdings für Lineare Algebra verwendet werden.
- Lineare Algebra ist die Sprache (linearer) statistischer Modelle.

Erzeugung

Die **matrix()** Funktion befüllt Matrizen mit Vektorelementen

- matrix(data, nrow, ncol, byrow)

```
matrix(c(1:12), nrow = 3)           # 3 x 4 Matrix der Zahlen 1,...,12, byrow = F
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

matrix(c(1:12), ncol = 4)           # 3 x 4 Matrix der Zahlen 1,...,12, byrow = F
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

matrix(c(1:12), nrow = 3, byrow = T)  # 3 x 4 Matrix der Zahlen 1,...,12, byrow = T
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Erzeugung

Die Funktion **cbind()** konkateniert passende Matrizen spaltenweise

```
A = matrix(c(1:4) , nrow = 2)      # 2 x 2 Matrix der Zahlen 1,...,4
     [,1] [,2]
[1,]    1    3
[2,]    2    4


B = matrix(c(5:10), nrow = 2)      # 2 x 3 Matrix der Zahlen 5,...,10
     [,1] [,2] [,3]
[1,]    5    7    9
[2,]    6    8   10


C = cbind(A,B)                     # spaltenweise Konkatenierung von A und B
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Erzeugung

Die Funktion **rbind()** konkateniert passende Matrizen reihenweise

```
A = matrix(c(1:6) , nrow = 2, byrow = T)  # 2 x 3 Matrix der Zahlen 1,...,6
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6


B = matrix(c(7:9), nrow = 1)          # 1 x 3 Matrix der Zahlen 5,...,10
     [,1] [,2] [,3]
[1,]    7    8    9


C = rbind(A,B)                        # reihenweise Konkatenierung von A und B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Charakterisierung

typeof() gibt den elementaren Datentyp einer Matrix aus

```
A = matrix(c(T,T,F,F), nrow = 2)      # 2 x 2 Matrix von Elementen vom Typ logical
typeof(A)
[1] "logical"

B = matrix(c("a","b","c"), nrow = 1)  # 1 x 3 Matrix von Elementen vom Typ character
typeof(B)
[1] "character"
```

nrow() und **ncol()** geben die Zeilen- bzw. Spaltenanzahl aus

```
C = matrix(1:12, nrow = 3)            # 3 x 4 Matrix
nrow(C)                               # Anzahl Zeilen
[1] 3
ncol(C)                               # Anzahl Spalten
[1] 4
```

Indizierung

Generell gilt

- Matrixelemente werden mit einen Zeilenindex und einem Spaltenindex indiziert.
- Die Indexreihenfolge ist immer 1. Zeile, 2. Spalte.
- Die Prinzipien der Indizierung entsprechen der Vektorindizierung.
- Indizes verschiedener Dimensionen können unterschiedlich indiziert werden.
- Eindimensionale Resultate liegen als Vektor, nicht als Matrix vor.

```
A = matrix(c(2:7)^2, nrow = 2)     # 2 x 3 Matrix der Zahlen 2^2,...,7^2
     [,1] [,2] [,3]
[1,]    4   16   36
[2,]    9   25   49

a_13  = A[1,3]                     # Element in 1. Zeile, 3. Spalte von A [36]
a_22  = A[2,2]                     # Element in 2. Zeile, 2. Spalte von A [35]
a_2.  = A[2,]                      # Alle Elemente der 2. Zeile [9,25,49]
a_.3  = A[,3]                      # Alle Elemente der 3. Spalte [36,49]
A_12  = A[1:2,1:2]                 # Submatrix der ersten zwei Zeilen und Spalten
A10   = A[A>10]                    # Elemente von A groesser 10 [16,25,36,49]
A_13  = A[1,c(F,F,T)]              # Element in 1. Zeile, 3. Spalte von A [36]
```

Arithmetik

Unitäre arithmetische Operatoren und Funktionen werden elementweise ausgewertet

```
A = matrix(c(1:4), nrow = 2)        # 2 x 2 Matrix der Zahlen 1,2,3,4
     [,1] [,2]
[1,]    1    3
[2,]    2    4

B = A^2                             # B[i,j]  = A[i,j]^2,  1 <= i,j <= 2
     [,1] [,2]
[1,]    1    9                      # 1^2, 3^2
[2,]    4   16                      # 2^2, 4^2

C = sqrt(B)                         # C[i,j]  = sqrt(A[i,j]^2),  1 <= i,j <= 2
     [,1] [,2]
[1,]    1    3                      # sqrt(1^2), sqrt(3^2)
[2,]    2    4                      # sqrt(2^2), sqrt(4^2)

D = exp(A)                          # D[i,j] = exp(A[i,j]),  1 <= i,j <= 2
     [,1] [,2]
[1,]  2.7 20.0                      # exp(1), exp(3)
[2,]  7.4 54.6                      # exp(2), exp(4)
```

Arithmetik

Matrizen passender Größe können mit binären arithmetischen Operatoren verknüpft werden

Binäre arithmetische Operatoren +,-,*,\werden bei gleicher Größe elementweise ausgewertet

```
A = matrix(c(1:4), nrow = 2)        # 2 x 2 Matrix der Zahlen 1,2,3,4
     [,1] [,2]
[1,]    1    3
[2,]    2    4

B = matrix(c(5:8), nrow = 2)        # 2 x 2 Matrix der Zahlen 5,6,7,8
     [,1] [,2]
[1,]    5    7
[2,]    6    8

C = A + B                           # C[i,j] = A[i,j] + B[i,j], 1 <= i,j <= 2
     [,1] [,2]
[1,]    6   10                      # 1 + 5, 3 + 7
[2,]    8   12                      # 2 + 6, 4 + 8

D = A * B
     [,1] [,2]
[1,]    5   21                      # 1 * 5, 3 * 7
[2,]   12   32                      # 2 * 6, 4 * 8
```

Arithmetik

Mit R Matrizen kann Lineare Algebra betrieben werden

- Addition, Subtraktion, Hadamardprodukt elementweise definiert wie oben

Multiplikation, Transposition, Inversion, Determinante

```
C = A  %*% B                    # 2 x 2 Matrixprodukt
      [,1] [,2]
[1,]   23   31                  # 1*5 + 3*6, 1*7+3*8
[2,]   34   46                  # 2*5 + 4*6, 2*7+4*8

A_T = t(A)                      # Transposition von A
      [,1] [,2]
[1,]    1    2                  # A[1,1], A[2,1]
[2,]    3    4                  # A[1,2], A[2,2]

A_inv = solve(A)                # Inverse von A
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5

A_det = det(A)                  # Determinante von A
[1] -2                          # 1*4 - 2*3
```

Attribute

Formal sind Matrizen atomic vectors mit einem **dim** Attribut

```
A = matrix(1:12, nrow = 4 )          #  4 x 3 Matrix
attributes(A)                        # Aufrufen der Attribute von A
$dim                                 # Schluessel
[1] 4 3                              # Wert: Anzahl Zeilen, Anzahl Spalten
```

rownames() und **colnames()** spezifizieren das Attribut **dimnames**

```
rownames(A) = c("P1","P2","P3","P4")  # Benennung der Zeilen von A
colnames(A) = c("Age", "Hgt", "Wgt")  # Benennung der Spalten von A
A                                     # A mit Attribut dimnames
   Age Hgt Wgt
P1   1   5   9
P2   2   6  10
P3   3   7  11
P4   4   8  12
attr(A, "dimnames")                   # Aufrufen des Attributs dimnames
[[1]]
[1] "P1" "P2" "P3" "P4"
[[2]]
[1] "Age" "Hgt" "Wgt"
```

Bei Matrizen ist die Benennung von Zeilen und Spalten eher ungewöhnlich.

Vektoren, Matrizen, Arrays, Faktoren

- Vektoren
- Matrizen
- **Arrays**
- Faktoren
- Übungen und Selbstkontrollfragen

Übersicht

- Arrays sind d-dimensionale, hyperrechteckige Datenstrukturen.
- Arrays sind die Generalisierung von Matrizen auf mehr als zwei Dimensionen.
- Für $d = 1$ entspricht ein Array einem Vektor, für $d = 2$ einer Matrix.
- Die Elemente eines Arrays sind vom gleichen Typ.
- $d_i, i = 1, ..., d$ ist die maximalen Anzahl von Elementen der $i$ten Dimension.
- Jedes Element eines Arrays hat einen d-dimensionalen Index $i_1, i_2, ..., i_d$.
- Formal sind Arrays in R d-dimensional interpretierte atomic vectors.

Erzeugung

Die array() Funktion befüllt Arrays mit Vektorelementen

- array(data, dim,...)
- dim enkodiert den maximalen Index in jeder der length(dim) Dimensionen

```
A = array(1:12, dim = c(2,2,3))        # 2 x 2 x 3 Array der Zahlen 1,...,12
, , 1
     [,1] [,2]
[1,]    1    3
[2,]    2    4

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

, , 3
     [,1] [,2]
[1,]    9   11
[2,]   10   12
```

- Es ist sinnvoll, sich von “räumlichen Vorstellungen” höherdimensionaler Arrays zu lösen.
- Arrays sind Datencontainer, Elemente werden durch Indexkombinationen addressiert.

Charakterisierung

length() gibt die Anzahl der Elemente eines Arrays aus

```
A = array(1:12, dim = c(2,2,3))          # 2 x 2 x 3 Array der Zahlen 1,...,12
length(A)                                # Die Anzahl der Elemente des Arrays
[1] 11                                   # ... ist 11
```

typeof() gibt den elementaren Datentyp eines Arrays aus

```
typeof(A)                                # Der Typ des Arrays
[1] "double"                             # ... ist double
```

dim() gibt die Dimensionen eines Arrays aus

```
dim(A)                                   # Dimension des Arrays
[1] 2 2 3                                # ... 2 x 2 x 3
```

Indizierung

Arrayindizierung erfolgt analog zu Vektor- und Matrixindizierung

```
A      = array(1:12, dim = c(2,2,3))   # 2 x 2 x 3 Array der Zahlen 1,...,12
a_223 = A[2,2,3]                       # Arrayelement mit Indexaddresse [2,2,3] (12)
```

Attribute

Formal sind Arrays atomic vectors mit einem **dim** Attribut

```
A = array(1:12, dim = c(2,2,3))        # 2 x 2 x 3 Array der Zahlen 1,...,12
attributes(A)                          # Aufrufen der Attribute von A
$dim                                   # Schluessel
[1] 2 2 3                              # Wert: Anzahl Zeilen, Anzahl Spalten
```

dimnames() kann zur Benennung der Arraydimensionen mit einer Liste benutzt werden.

Die Benennung von Arraydimensionen ist aber eher ungewöhnlich.

Arithmetik

Unitäre arithmetische Operatoren werden elementweise ausgewertet

Binäre arithmetische Operatoren werden bei Arraykonformität elementweise ausgwertet

```
A     = array(1:4, dim = c(1,2,2))    # 1 x 2 x 2 Array der Zahlen 1,...,4
B     = array(5:8, dim = c(1,2,2))    # 1 x 2 x 2 Array der Zahlen 5,...,8
C     = A^2                           # elementweise Potenzierung
, , 1
     [,1] [,2]
[1,]    1    4                        # 1^2, 2^2
, , 2
     [,1] [,2]
[1,]    9   16                        # 3^2, 4^2
D     = A + B                         # elementweise Addition
, , 1
     [,1] [,2]
[1,]    6    8                        # 1+5, 2+6
, , 2
     [,1] [,2]
[1,]   10   12                        # 3+7, 4+8
```

Vektoren, Matrizen, Arrays, Faktoren

- Vektoren
- Matrizen
- Arrays
- **Faktoren**
- Übungen und Selbstkontrollfragen

Übersicht

- Faktoren sind Vektoren, die nur vorab festgelegte Werte enthalten können.
- Faktoren können zum Speichern kategorialer Daten benutzt werden.
- Formal basieren Faktoren auf integer Vektoren.
- Faktoren haben die Attribute class “factor” und levels.
- Faktoren werden häufig beim Einlesen von Daten automatisch generiert.

Erzeugung

Faktoren können mit factor() aus Vektoren erzeugt werden.

- factor(x = character(), {levels, labels = levels, exclude = NA}, ...)

```
x = factor(c("a", "b", "b", "a"))   # Umwandlung eines character vectors
[1] a b b a                         # Inhalt des factors x
Levels: a b                         # Levels des factors x

y = factor(c(1,2,1,2,1,1))          # Umwandlung eines numerischen vectors
[1] 1 2 1 2 1 1                     # Inhalt des factors y
Levels: 1 2                         # Levels des factors y

v = c("a","b","b","a")              # Vektor zur Konversion in factor
z = factor(v,levels=c("a","c"))     # andere Levels als Werte in Vektor
[1] a    <NA> <NA> a                # fehlende Werte bei "b"
Levels: a c                         # Levels des factors z
```

Indizierung

Faktorindizierung geschieht analog zur Vektorindizierung

Attribute

Durch das **class** Attribut verhalten sich Faktoren nicht wie integer Vektoren

```
f = factor(c("f", "m", "m", "f"))   # Faktordefinition

attributes(f)                       # Attribute
$levels                             # Faktorlevels
[1] "f" "m"                         # f und m
$class                              # class Attribut
[1] "factor"                        # "factor"

typeof(f)                           # elementarer Datentyp
[1] "integer"                       # ... ist integer
```

Vektoren, Matrizen, Arrays, Faktoren

- Vektoren
- Matrizen
- Arrays
- Faktoren
- **Übungen und Selbstkontrollfragen**

Übungen und Selbstkontrollfragen

1. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle in einem kommentierten R Skript.
2. Beschreiben Sie in einer Übersicht die R Datenstruktur " Atomic Vector".
3. Beschreiben Sie in einer Übersicht die R Datenstruktur "Matrix".
4. Beschreiben Sie in einer Übersicht die R Datenstruktur "Array".
5. Beschreiben Sie in einer Übersicht die R Datenstruktur "Faktor".
6. Nennen Sie die R Befehle zum Erzeugen eines Vektors, einer Matrix, und eines Arrays.
7. Erzeugen Sie einen Vektor der Dezimalzahlen 0.0, 0.05, 0.10 , 0.15, ..., 0.90, 0.95, 1.0.
8. Wählen Sie mithilfe positiver Indices die Elemente 0.0, 0.1,...,0.9, 1.0 dieses Vektors aus.
9. Wählen Sie mithilfe negativer Indices die Elemente 0.0, 0.1,...,0.9, 1.0 dieses Vektors aus.
10. Wählen Sie die letzten 10 Elemente dieses Vektors aus.
11. Erzeugen Sie eine 4 x 5 Matrix M, deren Elemente dem Zeilenindex des Elements entsprechen.
12. Wählen Sie die 2. und 4. Zeile von M mithilfe positiver Indices aus.
13. Erzeugen Sie eine 4 x 5 Matrix N, deren Elemente dem Spaltenindex des Elements entsprechen.
14. Wählen Sie die 2. und 4. Zeile von N mithilfe negativer Indices aus.
15. Addieren Sie das Doppelte von M zu N.
16. Erzeugen Sie eine 4 x 5 Matrix, deren Elemente den Zeilen- und Spaltenindices entsprechen.

Literatur

Cotton, R. (2013). *Learning R*. O'Reilly, Beijing ; Sebastopol, CA, first edition edition.

Wickham, H. (2019). *Advanced R, Second Edition*. CRC Press.