Programmieren mit R für Einsteiger

3. Tabellen / 3.2 Matrizen

Berry Boessenkool 2022-02-25 11:40

Matrizen erstellen

```
matrix(data=1:6 , nrow=2, ncol=3)
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
m <- matrix(1:6 , nrow=2, ncol=3, byrow=TRUE)
m
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
dim(m) ;  nrow(m) ;  ncol(m) ;  length(m)
## [1] 2 3
## [1] 2
## [1] 3
## [1] 6
```

```
class(m) # zwei Klassen: array ist Überklasse von matrix
## [1] "matrix" "array"
```

Elementweise Multiplikation zweier Matrizen

```
m
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
m * 2   # Multiplikation per Element
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
```

```
n <- matrix(rep(0:1,each=3), ncol=3)   ;   n
##      [,1] [,2] [,3]
## [1,]    0    0    1
## [2,]    0    1    1
```

```
m * n                              # pro Element, auch für +,-, etc
##      [,1] [,2] [,3]
## [1,]    0    0    3
## [2,]    0    5    6
```

Zeilen- und Spaltennamen, Manipulation

```
colnames(m) <- c("A", "B", "C")    # wie bei data.frames
rownames(m) <- c("row1", "row2")
```

```
m[1,1] <- c(989) # Matrix ändern (manipulation)
```

```
m
##        A B C
## row1 989 2 3
## row2   4 5 6
```

Matrizen haben einen einzigen Datentyp für das ganze Objekt

Wird ein Element zu einer Zeichenkette geändert, werden alle konvertiert:

```
m[1,1] <- "a"
m
##      A   B   C
## row1 "a" "2" "3"
## row2 "4" "5" "6"
```

Nicht nur Spalten werden als Vektor ausgegeben (wie bei data.frames), sondern auch Zeilen:

```
m["row1", ]
##   A   B   C
## "a" "2" "3"
```

```
class(m["row1", ])                          # -> vector
## [1] "character"
is.vector(m["row1", ])          # nur 1 Datentyp -> downcast
## [1] TRUE
```

Funktion anwenden auf Zeilen / Spalten einer Matrix I

```
m <- matrix(1:12, ncol=4)  ;  m
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
rowSums(m) # + colSums. Achtung: rowsum() ist was anderes
## [1] 22 26 30
```

```
colMeans(m) # siehe auch rowMeans
## [1]  2  5  8 11
```

```
# Funktion 'median' auf Spalten anwenden:
apply(m, MARGIN=2, median)
## [1]  2  5  8 11
```

Funktion anwenden auf Zeilen / Spalten einer Matrix II

```
m
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# MARGIN=1: x Dimension behalten:
apply(m, MARGIN=1, FUN=median)
## [1] 5.5 6.5 7.5
```

```
# Weitere Argumente, die an 'median' weitergegeben werden
apply(m, MARGIN=1, FUN=median, na.rm=TRUE)
## [1] 5.5 6.5 7.5
```

```
apply(m, 1, function(x) sum(x==2)) # anonyme Funktion
## [1] 0 1 0
```

```
apply(m, 1, FUN=function(x) cat(toString(x)," -  "))
## 1, 4, 7, 10  -  2, 5, 8, 11  -  3, 6, 9, 12  -
## NULL
```

Zusammenfassung

`matrix`: Tabelle eines einzigen Datentypes

- `matrix(nrow, ncol, byrow)`, `ncol`, `nrow`, `dim`, `length`
- Arithmetische Operationen erfolgen per Element
- `colnames`, `rownames`
- `rowMeans`, `colSums`, `apply(X=mat, MARGIN, FUN)`

Matrizen transponieren

über die Diagonale drehen / spiegeln (Zeilen und Spalten vertauschen)

```
m
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
t(m) # transponieren
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

Matrizenmultiplikation

```
m <- matrix(1:6 , nrow=2, ncol=3, byrow=TRUE)
m
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6

n <- matrix(rep(1:5,each=3), ncol=5)
n
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    1    2    3    4    5
## [3,]    1    2    3    4    5
```

```
m %*% n
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    6   12   18   24   30
## [2,]   15   30   45   60   75
```

```
?"%*%"   # für die Dokumentation Anführungsstriche setzen
```

Konvertieren

Data.frames können umgewandelt werden:

```
d <- data.frame(AA=5:8, BB=6:9)
class(d)
## [1] "data.frame"
```

```
dm <- as.matrix(d)    ;  dm
##      AA BB
## [1,]  5  6
## [2,]  6  7
## [3,]  7  8
## [4,]  8  9

class(dm)
## [1] "matrix" "array"
```