Matrizen

Matrizen in R

- Matrizen sind die Erweiterung von Vektoren auf zwei Dimensionen
- Wie Vektoren, können Matrizen nur einen *atomaren* Datentyp speichern (ansonsten erfolgt wie bei Vektoren eine *coercion*)
- Matrizen sind wiederum ein Spezialfall von *Arrays* (mehr als zwei Dimensionen)
- Zum Erstellen von Matrizen gibt es den Befehl `matrix`
- Beim Erstellen werden Matrizen per Default nach Spalten befüllt

Matrizen in R

```
m <- matrix(1:4) # nrow = 1, ncol = 1 by default
m
##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
## [4,]    4
m2 <- matrix(1:4, nrow = 2)
m2
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
m3 <- matrix(1:4, nrow = 2, byrow = TRUE)
m3
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
sapply(list(m, m2, m3), typeof)
## [1] "integer" "integer" "integer"
```

Beispiel 'exams'

Beispiel: Speichern der Punkte aus 5 Hausarbeiten einer Übungsmappe für 3 Studierende

```
s1 <- c(10, 19, 19, 15, 18) # Ergebnisse der 5 Hausarbeiten a 10, und 4x20 Punkte
s2 <- c(9, 20, 17, 14, 18)
s3 <- c(5, 10, 13, 20, 13)
exams <- matrix(c(s1, s2, s3), nrow = 3, byrow = TRUE) # Eine Reihe pro Student
exams
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   10   19   19   15   18
## [2,]    9   20   17   14   18
## [3,]    5   10   13   20   13
str(exams)
##  num [1:3, 1:5] 10 9 5 19 20 10 19 17 13 15 ...
```

Operationen auf Spalten/Zeilen

- `apply`: Wie `lapply`/`sapply` aber arbeitet auf Matrizen (man muss zwischen Zeilen oder Spalten wählen)
- Spezialisierte Funktionen für Spalten-/Zeilen Summen bzw. Mittelwerte

```
# Berechne zeilenweise die Summe
apply(exams, 1, sum) # Summe Punkte pro Student
## [1] 81 78 61

# Berechne spaltenweise den Mittelwert
apply(exams, 2, mean) # Mittlere Punktezahl aller Studenten pro Hausarbeit
## [1]  8.00000 16.33333 16.33333 16.33333 16.33333

rowSums(exams) # colSums auch möglich, aber hier nicht sinnvoll
## [1] 81 78 61
colMeans(exams) # rowMeans auch möglich, aber hier nicht sinnvoll
## [1]  8.00000 16.33333 16.33333 16.33333 16.33333
```

- Wenn Zeilen-Summen / Mittelwerte gebraucht werden, nutze `colSums`, `rowSums`, `colMeans` und `rowMeans` (sehr effizient/schnell)
- `apply` allgemeiner und kann mit beliebigen Funktionen verwendet werden

Namensgebung von Matrizen

- Wie Vektoren/Listen, können Matrizen Namen haben
- Weil Sie zweidimensional sind, gibt es Zeilen- und Spaltennamen

```
rownames(exams) # keine Zeilennamen
## NULL
colnames(exams) # keine Spaltennamen
## NULL
rownames(exams)  <- paste0("student", seq_len(nrow(exams)))
rownames(exams)
## [1] "student1" "student2" "student3"
colnames(exams) <- paste0("exam", seq_len(ncol(exams)))
colnames(exams)
## [1] "exam1" "exam2" "exam3" "exam4" "exam5"
exams
##          exam1 exam2 exam3 exam4 exam5
## student1    10    19    19    15    18
## student2     9    20    17    14    18
## student3     5    10    13    20    13
```

Indizierung

- Auch bei Matrizen erfolgt die Indizierung (Zugreifen auf einzelne Elemente) über `[ ]` **eckige** Klammern
- Für Matrizen übernimmt die Indizierungsfunktion zwei Argumente (Zeilen und Spalten)

```
exams[, 2] # Punkte bei der zweiten Hausarbeit
## student1 student2 student3
##       19       20       10
exams[2, ] # Punkte des zweiten Studierenden
## exam1 exam2 exam3 exam4 exam5
##     9    20    17    14    18
exams[, 2, drop = FALSE] # Matrix-Struktur erhalten
##          exam2
## student1    19
## student2    20
## student3    10
exams[2, , drop = FALSE] # Matrix-Struktur erhalten
##          exam1 exam2 exam3 exam4 exam5
## student2     9    20    17    14    18
exams[3, 4] # Punkte des dritten Studierenden bei der 4ten Hausarbeit
## [1] 20
exams[2:3, 1:2] # Zeilen 2,3 Spalten 1, 2
##          exam1 exam2
## student2     9    20
## student3     5    10
exams[[10]] # 10. Element der Matrix
## [1] 15
```

Indizierung über logische Abfragen

```
exams[exams >= 15]
## [1] 19 20 19 17 15 20 18 18

which(exams >= 15) # An welcher Stelle stehen Elemente >= 27 (wenn exams ein Vektor wäre)
## [1]  4  5  7  8 10 12 13 14

which(exams >= 15, arr.ind = TRUE) # gibt Zeile + Spalte der ausgefählten Elemente
##          row col
## student1   1   2
## student2   2   2
## student1   1   3
## student2   2   3
## student1   1   4
## student3   3   4
## student1   1   5
## student2   2   5
```

Indizierung über Namen

Benamte Matrizen können auch über Zeilen-/Spaltennamen indiziert werden

```
exams[c("student2", "student3"), ]
##          exam1 exam2 exam3 exam4 exam5
## student2     9    20    17    14    18
## student3     5    10    13    20    13

exams[, paste0("exam", c(1, 3))]
##          exam1 exam3
## student1    10    19
## student2     9    17
## student3     5    13
```

Indizierung

Wie bei Vektoren, führt die Indizierung außerhalb der "Grenzen" zu einem Fehler

```
exams[5, 8]
## Error in exams[5, 8]: subscript out of bounds

exams["student10", ]
## Error in exams["student10", ]: subscript out of bounds

exams[[100]]
## Error in exams[[100]]: subscript out of bounds

exams[100]
## [1] NA
```

Das Erzeugen von NAs bei Indizierung außerhalb der Grenzen kann manchmal genutzt werden, um Vektoren unterschiedlicher Länge zu vereinheitlichen. Siehe z.B. hier

Hinzufügen, Löschen, Ändern

- Funktioniert wie bei Vektoren, nicht wie bei Listen

```
# Hinzufügen
exams[, ncol(exams) + 1] <- c(3, 5, 20, 20, 27) # won't work -> see rbind and cbind
## Error in `[<-`(`*tmp*`, , ncol(exams) + 1, value = c(3, 5, 20, 20, 27)): subscript out of bounds

# Löschen
exams[1, ] <- NULL # won't work for matrices
## Error in exams[1, ] <- NULL: number of items to replace is not a multiple of replacement length

exams2 <- exams[-1, ]

# Ändern
exams[3, 1] <- 10
exams
##          exam1 exam2 exam3 exam4 exam5
## student1    10    19    19    15    18
## student2     9    20    17    14    18
## student3    10    10    13    20    13
```

Hinzufügen von Spalten/Zeilen

- Hinzufügen von Spalten mit `cbind` (*column bind*)
- Hinzufügen von Zeilen mit `rbind` (*row bind*)

```
# Ergebnisse von eines weiteren Studierenden werden eingetragen
exams <- rbind(exams, c(1, 2, 17, 5, 20)) # Länge 5 weil 5 Hausarbeiten
exams
##          exam1 exam2 exam3 exam4 exam5
## student1    10    19    19    15    18
## student2     9    20    17    14    18
## student3    10    10    13    20    13
##              1     2    17     5    20

# Ergebnisse der 6ten Hausarbeit (für max. 30 Punkte) werden für alle Studierenden eingetragen
exams <- cbind(exams, c(30, 27, 29, 19)) # Länge 4, weil 4 Studierende
exams
##          exam1 exam2 exam3 exam4 exam5
## student1    10    19    19    15    18 30
## student2     9    20    17    14    18 27
## student3    10    10    13    20    13 29
##              1     2    17     5    20 19
```

Anteil erreichter Punkte

```
as.vector(exams)
##  [1] 10  9 10  1 19 20 10  2 19 17 13 17 15 14 20  5 18 18 13 20 30 27 29 19

exams / rep(c(10, 20, 20, 20, 20, 30), each = nrow(exams))
##          exam1 exam2 exam3 exam4 exam5
## student1   1.0  0.95  0.95  0.75  0.90 1.0000000
## student2   0.9  1.00  0.85  0.70  0.90 0.9000000
## student3   1.0  0.50  0.65  1.00  0.65 0.9666667
##            0.1  0.10  0.85  0.25  1.00 0.6333333
```

Type conversion

Wie bei Vektoren, führt das Zusammenfügen von Matrizen/Vektoren unterschiedlicher Datentypen zu *type conversion*:

```
grades <- cut(rowSums(exams), c(0, 60, 80, 90, 100, 110, 120), labels = rev(LETTERS[1:6]))

exams2 <- cbind(exams, grades)

exams2
##          exam1 exam2 exam3 exam4 exam5    grades
## student1    10    19    19    15    18 30      6
## student2     9    20    17    14    18 27      5
## student3    10    10    13    20    13 29      4
##              1     2    17     5    20 19      2

typeof(exams2)
## [1] "double"
```

Helpful functions

```
nrow(exams)
## [1] 4
ncol(exams)
## [1] 6
length(exams) # number of elements in a matrix  = nrow * ncol
## [1] 24
dim(exams)
## [1] 4 6
str(exams)
##  num [1:4, 1:6] 10 9 10 1 19 20 10 2 19 17 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:4] "student1" "student2" "student3" ""
##   ..$ : chr [1:6] "exam1" "exam2" "exam3" "exam4" ...
head(exams, 2) # ersten Zeilen
##          exam1 exam2 exam3 exam4 exam5
## student1    10    19    19    15    18 30
## student2     9    20    17    14    18 27
tail(exams, 2) # letzte Zeilen
##          exam1 exam2 exam3 exam4 exam5
## student3    10    10    13    20    13 29
##             1     2    17     5    20 19
```