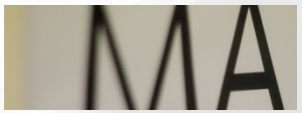
INSTITUT FÜR DEUTSCHE SPRACHE



Sandra Hansen-Morath
Sascha Wolfer

STATISTIK MIT R

Datentypen

Mitglied der Leibniz-Gemeinschaft

DATENTYPEN

- Ein Datentyp gibt an, welcher **Art** die Daten sind, die mit ihm beschrieben werden und welche **Operationen** auf diesen ausgeführt werden können.
- Grundlegende Unterscheidung:
  - **elementare Datentypen**: Zahl, Zeichenkette, Wahrheitswert
  - **komplexe Datentypen**: enthalten elementare Datentypen (und ggf. wiederum komplexe Datentypen)

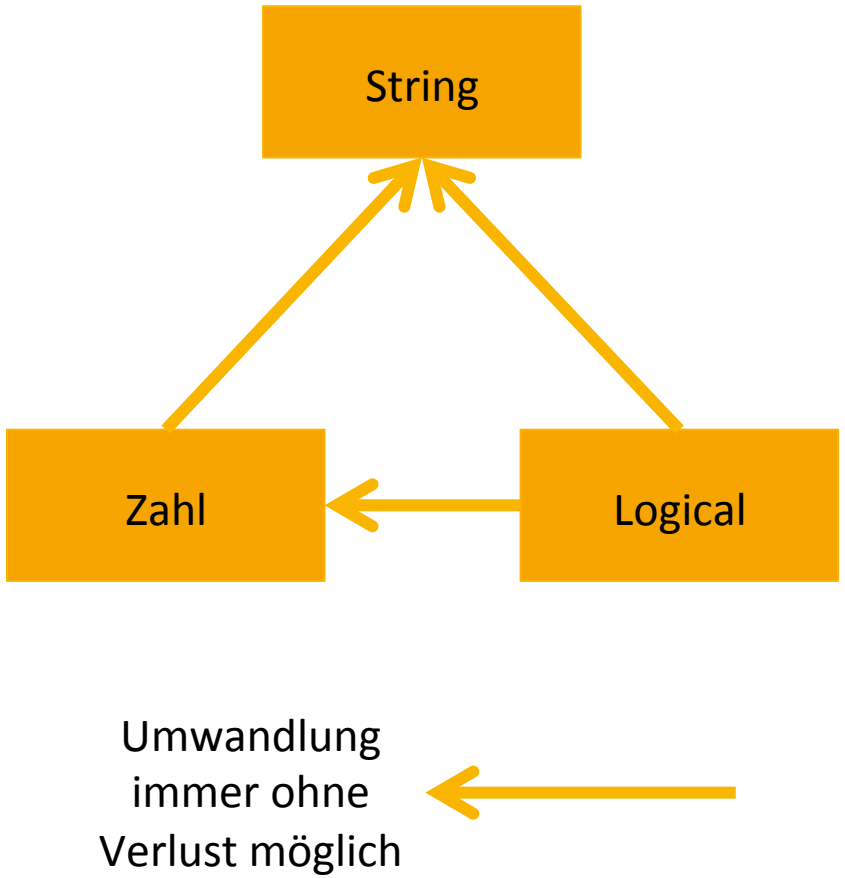
http://wirtschaftslexikon.gabler.de/Definition/datentyp.html

ELEMENTARE DATENTYPEN IN R

- **Zahlen**
  - Ganzzahlen/Integer
  - Kommazahlen
  - Beispiele: `42, 0.5, 2/3, -99, 0, Inf, -Inf, pi`
- **Zeichenketten/Strings**
  - der allgemeinste Datentyp
  - Beispiele: `"hallo welt", "", " ", "\t", "42"`
- **Wahrheitswerte/Logicals**
  - wahr oder falsch: TRUE oder FALSE
  - kann mit T und F abgekürzt werden (**nicht** `t` und `f`)

UMWANDLUNG VON ELEMENTAREN DATENTYPEN

- Elementare Datentypen können ineinander umgewandelt werden. Aber nicht beliebig.
- Alles kann in Strings umgewandelt werden (allgemeinster Datentyp).
- In einigen Fällen sind auch die anderen Typen ineinander umwandelbar.
  - bspw. "42" → 42
- Achten Sie auf die Warnmeldung "NAs durch Umwandlung erzeugt"!



UMWANDLUNGSFUNKTIONEN

- Umwandeln in Zahlen: `as.numeric()`
- Umwandeln in Ganzzahlen: `as.integer()`
- Umwandeln in Strings: `as.character()`
- Umwandeln in Wahrheitswerte: `as.logical()`

KOMPLEXE DATENTYPEN

- **Vektoren** enthalten Elemente *eines elementaren Typs*.
  - Konstruktion über `c()`
  - Zugriff über `[]`
- **Matrizen** enthalten Elemente eines elementaren Typs in 2 Dimensionen, typischerweise enthalten sie Zahlen.
  - Konstruktion über `matrix()`
  - Zugriff über `[,]`
- **Listen** enthalten beliebige andere Datentypen.
  - Konstruktion über `list()`
  - Zugriff über `[]`, `[[]]` und `$`

VEKTOREN

- Vektoren sind **der** grundlegende Datentyp in R.
- Vektoren sind geordnet, Zählung beginnt bei 1.
- Möglichkeiten zur Konstruktion von Vektoren:
  - combine: `c()`
  - x bis y (ganzzahlig): Doppelpunkt `x:y`
  - eigene Sequenzen: `seq(<von>, <bis>, <Schrittweite>)`
  - Wiederholen/replicate: `rep()`
  - und viele mehr...

FAKTOREN

- Ein Faktor ist eine spezielle Art von Vektor.
- Deklariert man etwas als Faktor, weist man R an, diesen Vektor als eine nominalskalierte Variable zu behandeln.
- In der Standardeinstellung von R werden alle eingelesenen Strings automatisch als Faktor kodiert. (Option stringsAsFactors)
- Faktoren haben Stufen (Levels), die mit ausgegeben werden.
  - `as.factor(c(4, 5, 2, 1, 1, 1, 3))`

DATAFRAMES

- Der wohl wichtigste komplexe Datentyp für die tägliche Arbeit ist der **Dataframe**.
- Dataframes enthalten Vektoren, die unterschiedlichen Typs sein können.
- Die Vektoren (Variablen) "stehen nebeneinander" und bilden so eine Datentabelle, wie wir sie bspw. aus Excel oder SPSS kennen.
  - Konstruktion: `data.frame()`
  - Zugriff über `[,]` und `$`

DATAFRAMES BILDLICH

	Wort	Länge	POS	logFreq	InhWort
1	Der	3	ART	5.42	F
2	Mann	4	NN	4.69	T
3	geht	4	VVFIN	4.59	T
4	gerne	5	ADV	3.63	T
5	lange	5	ADJA	3.98	T
6	Strecken	8	NN	2.96	T
7	.	1	$.	6.70	F
	Character	numerisch	Character	numerisch	logisch

DATAFRAMES KONSTRUIEREN UND EINLESEN

- Man kann Dataframes natürlich "von Hand" zusammenbauen.
- Typischerweise sind sie aber Produkt einer Einlese-Aktion.
- Häufigste Einlesevarianten in R:
  - `read.table()` oder Derivate: Daten aus CSV-Dateien einlesen
  - `load()` aus gespeicherten R-Objekten (`.Rdata`)
  - `gdata::read.xls()` aus Excel-Tabellen
  - `foreign::read.spss()` aus SPSS-Tabellen
- Unser Beispiel:
  - ```
    read.table("http://www.wolferonline.de/satz-df.tsv",
               sep = "\t", header = T)
    ```
- Einlesen mit Dateiauswahlfenster:
  `read.table(file.choose(), ...)`
    ```

INDIZIERUNG: ZUGRIFF AUF TEILMENGEN

- R bietet grundsätzlich **2 Arten der Indizierung**
  - Beispiel: `vec <- c("a", "b", "c", "d", "e", "f")`
  - über die **Nummern der Indizes**:
    - „Gib mir das 2 bis 5 Element.“: `vec[2:5]`
  - über **Wahrheitswerte**:
    - „Gib mir das 1. Element nicht, gib mir das 2. Element, gib mir das 3. Element, gib mir das 4. Element, gib mir das 5. Element, gib mir das 6. Element nicht.“: `vec[c(F, T, T, T, T, F)]`
- Der zweite Weg scheint zu kompliziert zu sein, gibt uns aber sehr mächtige Möglichkeiten zur Indizierung.
- Typischerweise schreibt man nämlich die Wahrheitswerte nicht selbst aus, sondern lässt sie von R berechnen!
- Das geschieht mit **Prädikaten**.

PRÄDIKATE

- Funktionen und Operatoren, die einen Wahrheitswert zurückgeben, nennt man Prädikate.
- Prädikate für Zahlen:
  - gleich: ==
  - größer/kleiner: > / <
  - größer gleich / kleiner gleich: >= / <=
  - ungleich: !=
- Prädikate können mit logischen Operatoren verknüpft werden:
  - logisches UND: &
  - logisches ODER: |
  - "nicht": !

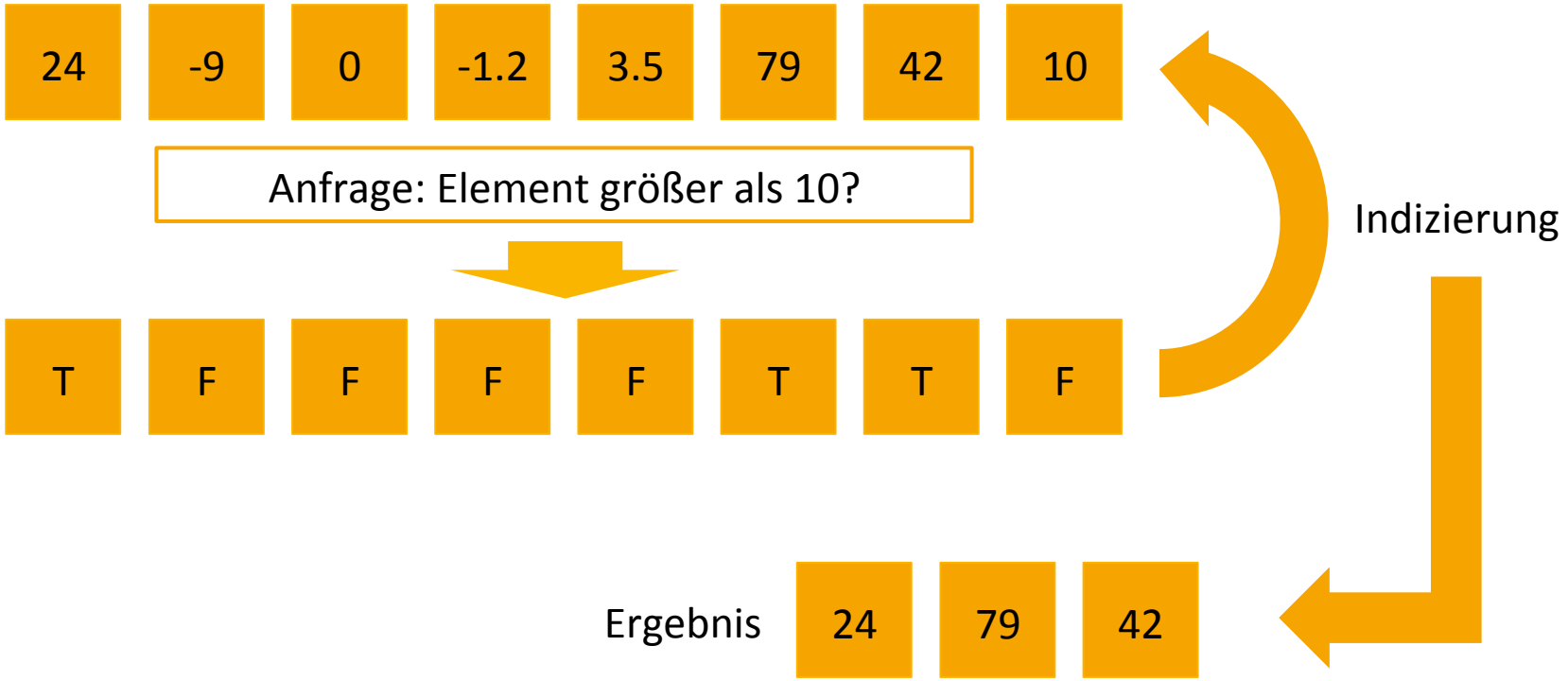
PRÄDIKATE

- Weitere nützliche Prädikate:
  - Ist x ein fehlender Wert? `is.na(x)`
  - Ist x in der Menge y? `x %in% y`

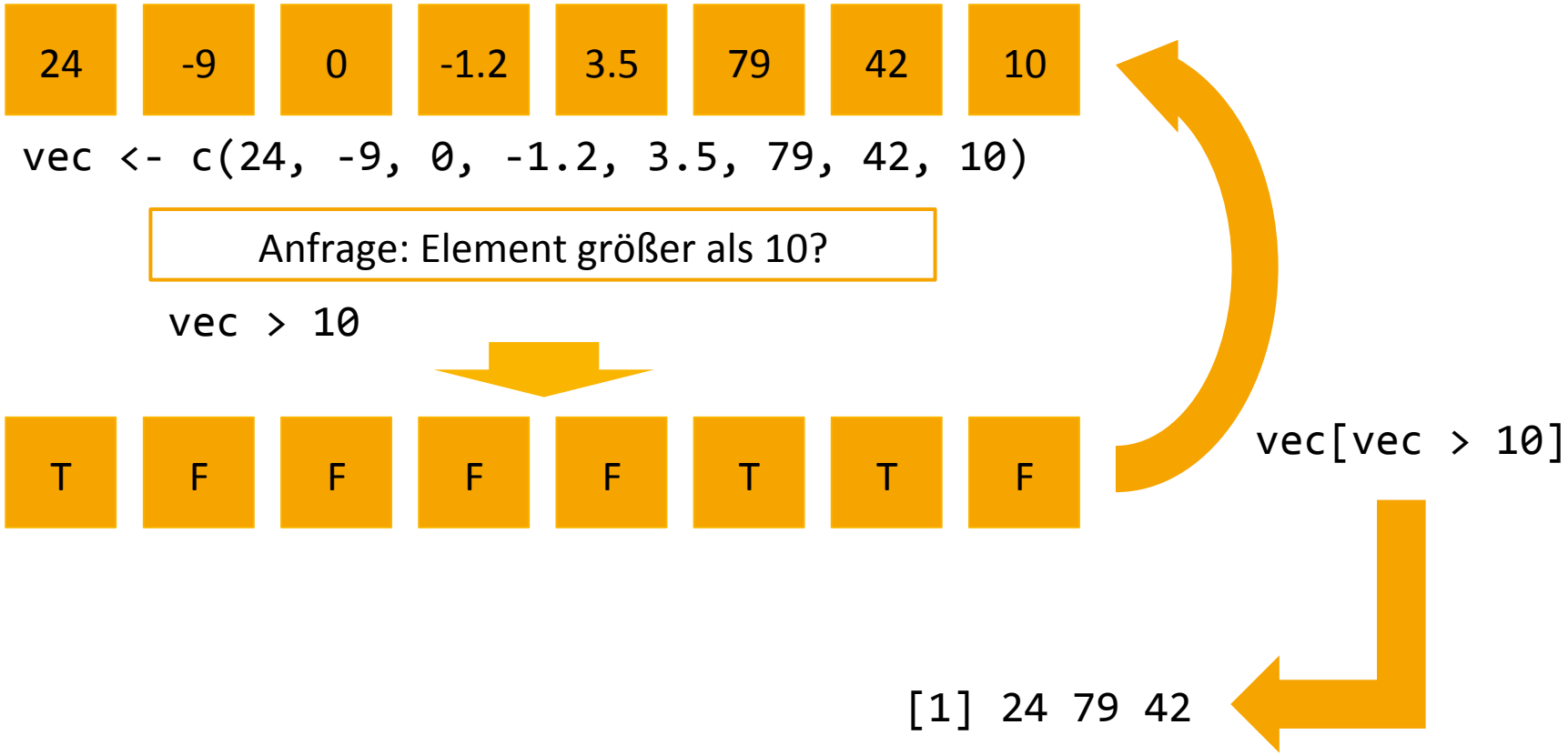
```
> artikel <- c("der", "die", "das")
> c("der", "mann", "das") %in% artikel
```

PRÄDIKATE UND INDIZIERUNG

- Verbinden wir solche Prädikate mit einer Indizierung, können wir gezielt einzelne Fälle aus einem Vektor herausgreifen.



PRÄDIKATE UND INDIZIERUNG IN R



DATAFRAMES INDIZIEREN

- Ganze Spalten können über $ abgerufen werden.
- Es können aber auch bestimmte Fälle ausgewählt werden.
- Die Indizierung von Dataframes kann in der allgemeinen Form dargestellt werden (Dataframe-Name: `dat`):

```
dat[<Zeilenrestriktion>,<Spaltenrestriktion>]
```

- Die Restriktionen sind **Beschränkungen**, die für selektierte Elemente gelten sollen.

DATAFRAMES INDIZIEREN

- Einlesen von "zeit.csv":
  - ```
    df <- read.table(file.choose(), sep = "\t", header = T,
                     fileEnc = "UTF-8")
    ```
- `df$wort`
- `df[df$laenge > 10, "wort"]`
- `df[df$laenge > 10 & df$freq > 10000, ]`
- `df[df$wort == "der" | df$wort == "Der", ]`

- Das Komma zwischen der Zeilen- und Spaltenrestriktion darf nicht vergessen werden.
- Mit logischen Verknüpfungen über UND und ODER können beliebig komplexe Abfragen formuliert werden.
  - Klammerung möglich
- Indiziert man einen Dataframe, bleiben die Zeilennummern erhalten.
    ```

DATAFRAMES INDIZIEREN

- Will man immer komplette Zeilen (= Fälle) extrahieren, kann man auch auf die Funktion `subset` zurückgreifen.

- `df[df$laenge > 20 & df$dok.freq > 1,]`
- `subset(df, laenge > 20 & dok.freq > 1,]`

- `subset` spart in manchen Fällen Tipparbeit und ist etwas intuitiver.

INFORMATIONEN ÜBER DATAFRAMES

- `nrow(<Dataframe>)`: Anzahl Zeilen / Fälle
- `head(<Dataframe>)`: erste 6 Fälle
- `names(<Dataframe>)`: Variablennamen
- `str(<Dataframe>)`: Informationen über alle Spalten
- `table(<Dataframe>$<Spalte>)`: Häufigkeitsauszählung einer Spalte
- `summary(<Dataframe>$<Spalte>)`: Zusammenfassung einer Spalte

LISTEN

- Listen können als Elemente beliebige Datentypen enthalten.
- Listen können also wiederum Listen enthalten usw.
- Ein Listenelement wird über [[ ]] extrahiert.
- Bereiche (mehrere Elemente von Listen) werden über [ ] extrahiert.

ÜBUNGEN

- Lesen Sie die Datei "eu.players.Rdata" ein.
- Geben Sie `head(eu.players)` ein.
- Finden Sie alle Spieler für die gilt:
  - `league` muss gleich "BuLi" sein.
  - `position` muss "Mittelstürmer" sein.
  - `val.mill` muss größer 10 sein.
- Übersetzt heißt das: Alle Bundesliga-Mittelstürmer, die mehr als 10 Millionen wert sind (Stand Januar 2013).
  - Auf wie viele Spieler trifft das zu?

BEGRIFFE

- Datentyp
- elementarer Datentyp
- komplexer Datentyp
- Zahlen
- Strings/Zeichenketten/Character
- Logicals/Wahrheitswerte
- Vektoren
- Matrizen
- Listen
- Dataframes

- Indizierung/Zugriff/Selektion
- Prädikate
- Zeilenrestriktion
- Spaltenrestriktion