

Programmieren mit R für Einsteiger

3. Tabellen / 3.7 Listen, Daten- & Objekttypen



Berry Boessenkool



frei verwenden, zitieren

2022-04-27 01:09

Eine `list` kann pro Element ein beliebiges R Objekt enthalten:

```
beispiel <- list(Ersteller="Berry",  
                 Tabelle=data.frame(a=426:424, b=rexp(3)),  
                 Sequenz=303:297 )
```

```
beispiel
```

```
## $Ersteller  
## [1] "Berry"  
## $Tabelle  
##      a      b  
## 1 426 0.2667456  
## 2 425 0.9632938  
## 3 424 1.6303822  
## $Sequenz  
## [1] 303 302 301 300 299 298 297
```

```
str(beispiel) # siehe auch head, tail, summary  
## List of 3  
## $ Ersteller: chr "Berry"  
## $ Tabelle : 'data.frame': ^I3 obs. of 2 variables:  
## ..$ a: int [1:3] 426 425 424  
## ..$ b: num [1:3] 0.267 0.963 1.63  
## $ Sequenz : int [1:7] 303 302 301 300 299 298 297
```

```
beispiel[3]      # -> list (mit 1 Element)
## $Sequenz
## [1] 303 302 301 300 299 298 297
beispiel[[3]]   # -> Element selbst (Hier: Vektor 7 Zahlen)
## [1] 303 302 301 300 299 298 297
```

```
beispiel$Sequenz # mit Namen indizieren
## [1] 303 302 301 300 299 298 297
beispiel$NichtExistent # -> kein Fehler!!, sondern NULL
## NULL
```

```
lapply(beispiel, class) # class auf jedes Element anwenden
## $Ersteller
## [1] "character"
## $Tabelle
## [1] "data.frame"
## $Sequenz
## [1] "integer"
```

```
sapply(unname(beispiel), class) # simplifizieren
## [1] "character" "data.frame" "integer"
```

R hat verschiedene Datentypen: Zahlen wie `42.6`, Zeichenketten wie `"R ist toll"`, Komplexe Zahlen wie `8+4i`, Kategorien mit `as.factor(c("blau", "rot"))`, Logische oder Boolesche Werte `c(TRUE, FALSE)` und weitere.

Funktionen und Operatoren können nicht immer alle verarbeiten:

```
zahl <- "1.3"
2 * zahl           # Fehler, weil 'zahl' eine Zeichenkette ist
## Fehler in 2 * zahl: non-numeric argument to binary operator

2 * as.numeric(zahl)
## [1] 2.6
```

```
zahl           # 'zahl' ist unverändert eine Zeichenkette
## [1] "1.3"
is.numeric(zahl)
## [1] FALSE
mode(zahl) # intern noch präziser: typeof
## [1] "character"
```

Beim Zusammenfassen mehrerer Datentypen werden diese (ohne Warnmeldung) transformiert:

```
c(42, "67")  
## [1] "42" "67"
```

```
c(42, TRUE)  
## [1] 42 1
```

Weil ein Vektor in R immer aus einem einzigen Datentyp besteht (für alle Einträge).

Übersicht: Datentypen (in Änderungsreihenfolge, order of coercion)

`c(TRUE,7) -> 1,7` ; `c(7,"z") -> "7" "z"`

adv-r.had.co.nz/Data-structures

Beschreibung	Beispiel	<code>typeof</code>	<code>class</code>
Leere Menge	<code>NULL</code>	<code>NULL</code>	<code>NULL</code>
Nicht angegeben	<code>NA</code>	<code>logical</code>	<code>logical</code>
Wahrheitswerte	<code>c(T, F, FALSE, TRUE)</code>	<code>logical</code>	<code>logical</code>
Kategorie	<code>factor("Links")</code>	<code>integer</code>	<code>factor</code>
Ganze Zahlen	<code>4:6</code> ; <code>7L</code>	<code>integer</code>	<code>integer</code>
Kommazahlen	<code>8.7</code>	<code>double</code>	<code>numeric</code>
Komplexe Zahlen	<code>5+3i</code>	<code>complex</code>	<code>complex</code>
Zeichenkette	<code>"R ist toll"</code>	<code>character</code>	<code>character</code>
Datum	<code>as.Date("2020-06-26")</code>	<code>double</code>	<code>Date</code>
Uhrzeit	<code>Sys.time()</code>	<code>double</code>	<code>POSIXct</code>
Funktion	<code>ncol</code>	<code>closure</code>	<code>function</code>

`as.character(3.14)` konvertiert Datentypen ; `is.integer(4:6)` prüft.

`str` zeigt eine Abkürzung von `class`. Weitere seltene Typen: `raw`, `environment`, `promise`, ...

`mode` \approx `typeof`, aber `integer/double -> numeric` ; `closure/special/builtin -> function`.

Bei `date/time` bestimmt das zuerst auftretende die Output `class`.

Alles was in R existiert, ist ein Objekt (auch eine Funktion)

Objekte haben Klassen und dafür bestehen Methoden.

`str` z.B. zeigt unterschiedliche Ausgaben, abhängig der `class`.

```
v <- c(6.3,2) ; df <- data.frame(1:4,4:1) ; m <- matrix(1:6)
```

```
str(v)
```

```
## num [1:2] 6.3 2
```

```
str(df)
```

```
## 'data.frame': ^I4 obs. of 2 variables:
```

```
## $ X1.4: int 1 2 3 4
```

```
## $ X4.1: int 4 3 2 1
```

```
str(m)
```

```
## int [1:6, 1] 1 2 3 4 5 6
```

```
str(append)
```

```
## function (x, values, after = length(x))
```

```
length(v) # für Vektoren
```

```
## [1] 2
```

```
colnames(df) # für Tabellen
```

```
## [1] "X1.4" "X4.1"
```

```
dim(m)
```

```
## [1] 6 1
```

```
which(v > 4) # für logicals
```

```
## [1] 1
```

Objekt	Beispiel	<code>typeof</code>	<code>class</code>
Vektor	<code>c(pi, 2)</code> siehe Datentypen	<code>...</code>	<code>...</code>
Matrix	<code>matrix(9:15, ncol=2)</code>	<code>...</code>	<code>matrix</code>
Array	<code>array(letters, dim=c(2,6,4))</code>	<code>...</code>	<code>array</code>
Tabelle	<code>data.frame(4:5, B=c("a", "b"))</code>	<code>list</code>	<code>data.frame</code>
Liste	<code>list(e11=7:15, e12="big")</code>	<code>list</code>	<code>list</code>
Funktion	<code>function(x) 12+0.5*x</code>	<code>closure</code>	<code>function</code>
...	<code>lm(b ~ a)</code>	<code>list</code>	<code>lm</code>

Eine `matrix` hat einen einzigen Datentyp. Wenn ein Element geändert wird, werden alle konvertiert (in **Änderungsreihenfolge**).

Ein `data.frame` kann mehrere Datentypen haben, einen pro Spalte.

Eine `list` kann alles kombinieren, auch eine weitere Liste.

`is.atomic(Objekt)` zeigt TRUE (**vector, matrix, array**) oder FALSE

`is.vector(Objekt)` zeigt TRUE sehr unerwartet

`as.matrix(Objekt)` konvertiert ein Objekt zwangsweise.

Listen:

- ▶ `list`, `str`, `summary` etc.
- ▶ `list$elementname`, `list[[index]]`
- ▶ `lapply`, `sapply(liste, funktion)`

Übersicht über Daten- und Objekt-typen:

- ▶ Datentypen: numeric (integer/double), character, logical, complex
- ▶ `class`, `mode`, `typeof`
- ▶ Objekttypen: vector, matrix, array, data.frame, list
- ▶ Übersichten

Für diese Lektion gibt es keine Übungsaufgaben

Vorsicht bei `is.vector`:

- FALSE für `factor` / `date` / `time` - TRUE für `list` siehe SO Frage

Nutze `is.atomic(x) && is.null(dim(x))`

Datentypen

	is.vector	is.atomic
null	FALSE	TRUE
na	TRUE	TRUE
logi	TRUE	TRUE
factor	FALSE	TRUE
int	TRUE	TRUE
double	TRUE	TRUE
complex	TRUE	TRUE
char	TRUE	TRUE
date	FALSE	TRUE
time	FALSE	TRUE
func	FALSE	FALSE

Objekttypen

	is.vector	is.atomic
vector	T/F	TRUE
matrix	FALSE	TRUE
array	FALSE	TRUE
data.frame	FALSE	FALSE
list	TRUE	FALSE
function	FALSE	FALSE
lm	FALSE	FALSE

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

Nicht verwenden:

```
if( class(m) == "matrix" ) message("m ist eine Matrix")
## Warning in if (class(m) == "matrix") message("m ist
eine Matrix"): the condition has length > 1 and only the
first element will be used
## m ist eine Matrix
```

verwenden:

```
if( inherits(m, "matrix") ) message("m ist eine Matrix")
## m ist eine Matrix
```