Programmieren mit R für Einsteiger

1. Grundlagen / 1.3 Vektoren

Berry Boessenkool 2022-02-25 11:40

Vektoren I

Vektoren in R sind keine geometrischen Konstrukte, sondern eine geordnete Menge. (ordered set of values).
Vektoren werden erstellt mit `c` (Combine / Concatenate). Einträge werden mit einem Komma getrennt.

```
zahlen <- c(3, 7, -2.7654321, 11, 3.8, 9)
```

Objekt aufrufen (anzeigen):

```
zahlen
## [1]  3.000000  7.000000 -2.765432 11.000000  3.800000
## [6]  9.000000
```

```
print(zahlen, digits=3) # Explizit anzeigen, mit Optionen
## [1]  3.00  7.00 -2.77 11.00  3.80  9.00
```

Vektoren II: Folgen

```
1:5  # Ganze Zahlen (integers)  von : bis
## [1] 1 2 3 4 5
```

```
rep(1:4, times=3) # Zahlen mehrfach wiederholen
##  [1] 1 2 3 4 1 2 3 4 1 2 3 4
```

```
rep(1:3, each=3, times=2)
##  [1] 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3
```

```
seq(from=3, to=-1, by=-0.5) # Sequenz
# Für absteigende Folgen muss 'by' negativ sein
## [1]  3.0  2.5  2.0  1.5  1.0  0.5  0.0 -0.5 -1.0
```

```
seq(1.32, 6.1, length.out=9) # 9 Elemente
## [1] 1.3200 1.9175 2.5150 3.1125 3.7100 4.3075 4.9050
## [8] 5.5025 6.1000
```

```
seq(1.32, 6.1, len=15) # Argumentnamen abkürzbar
```

Indexing: Submengen auswählen -> Eckige Klammern

```
vek <- c(3, 7, -2, 11, 4, 9)
```

```
vek[1] # Erstes Element zurückgeben
## [1] 3
```

AltGr + 8 / 9,
Option + 5 / 6

```
vek[2:4] # Mehrere Elemente auswählen
## [1]  7 -2 11
```

```
vek[ c(2,5,1,6,1) ] # Flexible Reihenfolge
## [1] 7 4 3 9 3
```

```
vek[-2] # Alle Elemente außer das zweite
## [1]  3 -2 11  4  9
```

```
vek[-(1:3)] # Alle Elemente außer den ersten drei
## [1] 11  4  9
```

```
vek[-1:3] # geht nicht
## Fehler in vek[-1:3]:  only 0's may be mixed with negative subscripts
-1:3      # weil -1 und 1 nicht beides erfüllt werden kann
## [1] -1  0  1  2  3
```

head/tail, str, class, length

```
a <- seq(from=1, to=100, by=0.1)
head(a) # Die ersten 6 Elemente anzeigen
## [1] 1.0 1.1 1.2 1.3 1.4 1.5
```

```
tail(a, 8) # Die letzten 8 Elemente
## [1]  99.3  99.4  99.5  99.6  99.7  99.8  99.9 100.0
```

```
a[2] <- 87 # Einzelnes Element eines Objekts ändern
head(a) # das Objekt 'a' ist jetzt anders
## [1]  1.0 87.0  1.2  1.3  1.4  1.5
```

```
str(a) # Struktur: Datentyp, [Dimension], erste Werte
##  num [1:991] 1 87 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 ...
```

```
class(a) # primär: numeric, logical, factor, character
## [1] "numeric"
```

```
length(a) # Länge (Anzahl Elemente) des Vektors
## [1] 991
```

Recycling: Vektoren nach Bedarf erweitern

```
2 * 7
## [1] 14
```

```
2:9 * 7          # 7 wird so oft wiederholt wie nötig
## [1] 14 21 28 35 42 49 56 63
```

```
2:9 * c(7,1)     # Dieses Konzept heißt "Recycling"
## [1] 14  3 28  5 42  7 56  9
```

```
2:9 * c(7,1,2) # Ergebnis mit Warnung, wenn's nicht passt
## Warning in 2:9 * c(7, 1, 2): longer object length is
not a multiple of shorter object length
## [1] 14  3  8 35  6 14 56  9
```

Zusammenfassung

Vektoren erstellen und indizieren:

- `c`, `:`, `rep`, `seq`
- `v[n]`, `v[-n]`, `v[m:n]`, `v[-(m:n)]`
- `head`, `tail`, `str`, `class`, `length`
- Recycling

Vektoren mit Namen

```
Punktestand <- c(Christoph=19, Berry=17, "Anna Lena"=22)
```

Leerzeichen in Namen besser vermeiden

```
Punktestand[2] # Index: Position
## Berry
##    17
```

```
Punktestand["Berry"] # Index: Name
## Berry
##    17
```

```
names(Punktestand) # 'Punktestand' ist ein "named vector"
## [1] "Christoph" "Berry"     "Anna Lena"
```

```
names(Punktestand) <- LETTERS[1:3]
names(Punktestand)[2] <- "NeuerName"
Punktestand
##         A NeuerName         C
##        19        17        22
```