Programmieren mit R für Einsteiger

3. Tabellen / 3.1 DataFrames

Berry Boessenkool 2022-02-25 11:40

Tabellarische Daten werden in R fast immer als `data.frame` angelegt. Jede Spalte kann einen eigenen Datentyp haben (numeric, character, factor, logical, etc).

```
?data.frame
```

```
bdf <- data.frame(Zahlen=11:14, Buchstaben=letters[1:4],
                  Booleans=(1:4)>2)
bdf # bdf: BeispielDataFrame
##   Zahlen Buchstaben Booleans
## 1     11          a    FALSE
## 2     12          b    FALSE
## 3     13          c     TRUE
## 4     14          d     TRUE
```

data.frames anschauen

Große Tabellen nicht komplett anzeigen (überfüllt die Console)
-> `str` und `summary` verwenden:

```
str(bdf)
## 'data.frame': 4 obs. of  3 variables:
##  $ Zahlen    : int  11 12 13 14
##  $ Buchstaben: chr  "a" "b" "c" "d"
##  $ Booleans  : logi  FALSE FALSE TRUE TRUE
```

```
summary(bdf) # Hilfreiche Info pro Spalte
##      Zahlen       Buchstaben         Booleans
##  Min.   :11.00   Length:4           Mode :logical
##  1st Qu.:11.75   Class :character   FALSE:2
##  Median :12.50   Mode  :character   TRUE :2
##  Mean   :12.50
##  3rd Qu.:13.25
##  Max.   :14.00
```

Untermenge (Subset): data.frame-Auswahl nach Position (Index)

Zum Indexing eckige Klammern verwenden (wie bei Vektoren), allerdings zwei Dimensionen angeben (kommagetrennt):

```
bdf[ 3 , 1 ] # Wert in der dritten Zeile, erste Spalte
## [1] 13
```

```
bdf[   , 2 ] # Alle Zeilen, zweite Spalte (-> Vektor)
## [1] "a" "b" "c" "d"
```

```
bdf[2, ] # Alle Spalten = komplette Zeile (-> data.frame)
##   Zahlen Buchstaben Booleans
## 2     12          b    FALSE
```

```
?"[" # für die Dokumentation über Subsetting
```

```
nrow(bdf)
## [1] 4
```

```
ncol(bdf) # Siehe auch dim(bdf)
## [1] 3
```

data.frame Auswahl mit Spaltennamen

```
bdf[,"Booleans"]
## [1] FALSE FALSE  TRUE  TRUE
```

Mit Rstudio autocompletion (TAB -taste): $ (Shift + 4)

```
bdf$Booleans
## [1] FALSE FALSE  TRUE  TRUE
```

```
colnames(bdf)
## [1] "Zahlen"     "Buchstaben" "Booleans"
```

```
colnames(bdf)[2] <- "Zeichen"    # Ändert das Objekt bdf
bdf
##   Zahlen Zeichen Booleans
## 1     11        a    FALSE
## 2     12        b    FALSE
## 3     13        c     TRUE
## 4     14        d     TRUE
```

Auswahl mehrerer Zeilen / Spalten

```
bdf[2:3, ] # Zeilen 2 bis 3
##   Zahlen Zeichen Booleans
## 2     12       b    FALSE
## 3     13       c     TRUE
```

```
bdf[c(4,1), ] # Vektoren zum Indizieren
##   Zahlen Zeichen Booleans
## 4     14       d     TRUE
## 1     11       a    FALSE
```

```
bdf[bdf$Buchstaben=="c", ] # Logische Werte: 'Filtern"
## [1] Zahlen   Zeichen  Booleans
## <0 rows> (or 0-length row.names)
```

```
bdf[-2, 2:1] # Negativ-auswahl wie bei Vektoren
##   Zeichen Zahlen
## 1       a     11
## 3       c     13
## 4       d     14
```

Spalten ändern / hinzufügen / löschen

```
# bestehende Spalte überschreiben:
bdf$Zahlen <- 45:48
# neue Spalte am Ende hinzufügen:
bdf$Zeit3000m <- c(12.08, 10.27, 11.79, 13.50)
```

```
bdf
##   Zahlen Zeichen Booleans Zeit3000m
## 1     45       a    FALSE     12.08
## 2     46       b    FALSE     10.27
## 3     47       c     TRUE     11.79
## 4     48       d     TRUE     13.50
```

```
# Spalte entfernen:
bdf$Zeichen <- NULL
```

```
bdf
##   Zahlen Booleans Zeit3000m
## 1     45    FALSE     12.08
## 2     46    FALSE     10.27
## 3     47     TRUE     11.79
## 4     48     TRUE     13.50
```

data.frame Tipps und Tricks

Spaltenauswahl

- `df[ ,2]` wählt immer die zweite Spalte aus
- `df[ ,"name"]` ist unabhängig der Spaltenreihenfolge und von der Lesbarkeit her besser verständlich (außer man weiß genau, was die zweite Spalte beinhaltet)
- `df$name` ist weniger Tipparbeit und kann mit Autocompletion ohne Tippfehler eingegeben werden

Generelle Tipps

- Wenn `nrow(df)` `NULL` zurückgibt, könnte `df` ein Vektor sein.
- `NROW(df)` zeigt `length(df)`, wenn `df` ein Vektor ist.
- Objekte konvertieren mit `as.data.frame(theMatrix)`
- Wenn Spaltennamen mit einer Zahl anfangen, wird der Prefix "X" oder "V" (Variable) vorgesetzt. Das kann besonders beim Einlesen von Daten passieren.

Zusammenfassung

Tabellen erstellen und indizieren:

- `data.frame`, `str`, `summary`
- `nrow`, `ncol`, `colnames`
- `df[r,c]`, `df[r,]`, `df[,c]`, `df[,"cname"]`, `df[,c("cn1","cn2")]`
- `$cname`, `$newCol <-`

Spalte als data.frame ausgeben


```
bdf[ , "Booleans"]                    # -> Vektor
## [1] FALSE FALSE  TRUE  TRUE
```

```
bdf[ , "Booleans", drop=FALSE ]  # -> data.frame
##   Booleans
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
```

```
bdf["Booleans"]          # Ohne Komma -> data.frame   ACHTUNG
```

Nicht nutzen! `objekt[index]` ist für Vektoren!
R kann das, aber der Leser deines Codes nicht.

Zeilennamen

```
rownames(bdf) <- c("Alex", "Berry", "Christoph", "Daniel")
bdf
##           Zahlen Booleans Zeit3000m
## Alex          45    FALSE     12.08
## Berry         46    FALSE     10.27
## Christoph     47     TRUE     11.79
## Daniel        48     TRUE     13.50
```

```
bdf["Berry", ]
##       Zahlen Booleans Zeit3000m
## Berry     46    FALSE     10.27
```

Zusammenfassung Zusatzmaterial

- `tabelle[, spalte, drop=FALSE]` wenn ein data.frame statt eines Vektors gewollt ist (Syntax `tabelle[spalte]` nicht nutzen)
- `rownames`