Einführung in R

Peter Nauroth

1. Was ist R?

Zentrale Strukturen

R ist eine Programmiersprache

- Datenstrukturen & Datentyp (z.B. Variablen)
  - Datentyp (`mode()`): numeric, character, boolean, factor
  - Datenstruktur (`str()`)
  - *Viele Fehlermeldungen sind darauf zurückzuführen, dass der Datentyp oder die Datennstruktur nicht zur Funktion passen!*
- Operatoren (z.B. Zuordnung: `x <- 5`)
- Funktionen (z.B. `'+'(4,5)`)
  - Funktionsparameter

2. RStudio

GUI

1. **Konsole**
   - Taschenrechner (`4+4`)
2. **Skript**
3. **Workspace / History / GIT**
   - *Viele Probleme sind auf unbeabsichtigte Objekte im Workspace zurükzuführen.*
     - `rm(list=ls())` –> Säubern des Workspace
     - erfordert mehr Mikromanagement als SPSS

4. Files / Plots / Packages /Help

- Working Directory beachten (`setwd()`)
- Hilfe (`?`; z.B. `?lm`)
  - Packages / Libraries
  - Für (fast) alles gibt es bereits vorgefertigte Funktionen in Packages
  - Um diese nutzen zu können:`get.packages()` & `library()`

3. Datenstrukturen

3.1 Homogene Datenstrukturen

Homogene Datenstrukturen enthalten nur einen Datentyp.

1. Skalare

```
s <- 5
s
str(s)
s + 4
s * 2
```

2. Vektoren

```
v <- c(5,4,5)
v
str(v)
v + 4
v * 2
# get 3rd element:
v[3]
```

3. Matrizen

```
m1 <- matrix(c(1,123,4,12,3,5), nrow=2)
m1
str(m1)
m1 + 4
m1 * 2
m2 <- matrix(c("1","123","4","12","3","5"), nrow=2)
m2
str(m2)
# get 2nd element in 1st row:
m2[1,2]
# get 2nd column:
m2[,2]
```

3.2 Heterogene Datenstrukturen

Datenstruktur mit potentiell unterschiedlichen Datentypen.

1. Data frames

```
subject <- c(2,3,4)
condition <- as.factor(c("EG","KG","EG"))
dv <-  c(5,5,5)
df <- data.frame(subject, condition, dv)
df
str(df)

# get values on dv:
df$dv
# alternatively
df[,3]
# get values of subject 3:
df[df$subject==3,]
# alternatively
df[2,]
```

3.2 Heterogene Datenstrukturen

2. Listen

```
l <- list(c("a", "b", "c"), c(1,2,3,4),
          c(TRUE, FALSE, TRUE))
l
str(l)

# get 1st value of 3rd object in list
l[[3]][1]
```

- Viele Outputs von statistischen Funktionen sind Listen.

Das Verständnis über den Zugriff auf und die Organisation von den verschiedenen Datenstrukturen ist essentiell für das Arbeiten mit R.

4. Funktionen

Grundsätzliches

Funktionen sind Methoden mit denen wir unsere Daten manipulieren und analyiseren.

- Ein Funktion hat einen *Namen* (z.B.: `mean`),
- *Parameter* bzw. *Argumente* (z.B.: `c(1,2,3)`) und
- einen *Rückgabewert* (z.B.: `## [1] 2`).

```
x <- c(1:10)
x
# Calculate mean of x
mean(x) # That's a function call and x is a function
        # parameter/argument
# However, `mean` may have more parameters. Try ?mean
```

TODO: Berechne den Mittelwert von `x <- c(1:78, NA)`.

Selbstdefinierte Funktionen

- Funktionen können selbst definiert werden:

```
# We want a function that squares its input:
x <- 5
x*x
f1 <- function(x) x*x
f1(x)
```

TODO1: Definiere eine Funktion, die ihren Input verdoppelt und anschließend 5 addiert.

TODO2: Definiere eine Funktion, die den Mittelwert eines Vektors berechnet und vorher mögliche NAs entfernt (Tipp: `?mean`).

5. Datenmanipulation