Technische Universität München

PROTECT Behavioral Health Promotion & Technology Lab

Evaluation Randomisiert-Kontrollierter Studien und Experimente mit R

R basics - Funktionen, Objekte, Operatoren und Fehlermeldungen

Prof. Dr. David Ebert & Mathias Harrer

Graduiertenseminar TUM-FGZ

Psychology & Digital Mental Health Care, Technische Universität München

Funktionen

R als “Taschenrechner”

Unabhängig von spezifischen Funktionen, können mithilfe von R klassische Rechenaufgaben gelöst werden:

Probieren Sie selbst:

```
5*28/7
(12+67-3)*4
```

Was sind Funktionen?

Funktionen sind Kernelemente von R: Sie erlauben es, (meist in Bezug auf unsere eigenen Daten) vordefinierte Operationen auszuführen.

Parallele zur mathematischen Formulierung einer Funktion *f(x)*:

→ In R wird eine Funktion definiert, indem erst der Name der Funktion und dahinter in Klammern input und/ oder weitere Spezifikationen für die Funktion (sog. Argumente) aufgeschrieben werden

```
Funktionsname(Argument1 = Wert1, Argument2 = Wert2, ...)
```

Position Matching

Der Agrumentname (z.B. ”Argument1 =”) kann auch weggelassen werden, wenn die Reihenfolge der Argumente eingehalten wird.

(Harrer et al., 2021, Kap. 2.2)

Beispiele (I)

1. Was ist die Wurzel von 9? Dazu können die Funktionen sqrt() nutzen:

```
sqrt(9)
## [1] 3
```

2. Logarithmus (log()) aller Werte der Variable “age” im Datensatz:

```
log(data$age)
## [1] 3.850148 3.737670 3.951244 3.135494 3.828641 ...
```

→ Statt eines konkreten Wertes wird die gesamte Variable in die Funktion eingespeist, indem die Variable ”age” über das Dollarzeichen aus unserem Datensatz ”data” ausgewählt wird.

Beispiele (II)

3. Mittelwert (mean()) der Variable “pss” zum Post-Zeitunkt:

```
mean(data$pss.1)
## [1] NA
mean(data$pss.1, na.rm = TRUE)
[1] 20.42387
```

→ Das Ergebnis der ersten Zeile Code ist ”not available”, da zum Post-Zeitpunkt Beobachtungen fehlen und der Mittelwert so nicht berechnet werden kann. Durch die Spezifikation des Arguments ”na.rm” als ”TRUE”, wird der Mittelwert nur über die beobachteten Werte gebildet und kann somit ausgegeben werden.

Tipps im Umgang mit Funktionen (I)

Info

Auch deutlich komplexere Funktionen in R funktionieren nach dem gleichen Prinzip: Man gibt die Parameterinformationen ein, die eine Funktion benötigt, und die Funktion nutzt diese Information, um ihre Berechnungen durchzuführen und schließlich das Ergebnis anzuzeigen.

(Harrer et al., 2021, Kap. 2.2)

Tipps im Umgang mit Funktionen (II)

Hilfe und R Documentation

- Manche Funktionen in R verlangen mehrere Agrumente und niemand kann die korrekte Nutzung aller Funktionen auswendig lernen
- Die Lösung: Detaillierte Beschreibungen der Funktionen in R Documentation
- Öffnen von R Documentation entweder über *Help* im rechten unteren Fenster in Rstudio und Eingabe der Funktion in die Suchleiste oder über das Ausführen von *?funktionsname* in der Konsole (z.B. ?mean)

(Harrer et al., 2021, Kap. 2.5)

Tipps im Umgang mit Funktionen (III)

Default Arguments

- Darunter werden Argumente einer Funktion verstanden, deren Wert vordefiniert ist und automatisch genutzt wird
- Default Arguments müssen beim Schreiben der Funktion also nur hinzugefügt werden, wenn sie explizit von den Voreinstellungen abweichen
- Default Werte einer Funktion können im Abschnitt ”Usage” in R Documentation eingesehen werden

(Harrer et al., 2021, Kap. 4.2)

Functional programming languages

Im Gegensatz zu *"object-oriented programming languages"* konzentrieren sich *"functional programming languages"* bei der Problemlösung auf Funktionen

Hauptmerkmale von functional programming languages

- **first-class functions:** Vielseitige Einsetzbarkeit von Funktionen
- **pure functions:** Output der Funktion hängt vom Input ab (d.h. Output reproduzierbar) und keine Nebeneffekte der Funktion (wie z.B. Veränderung des Wertes einer globalen Variable)

$\Rightarrow$ Auch wenn R diesen Kriterien nicht vollkommen entspricht, kann **R im Kern als functional programming language definiert** werden

Wickham (2019)

Objekte

Neue Objekte in R erstellen (I)

Zum Speichern von Werten (z.B. Zahlen) in Objekten kann der Zuweisungsoperator <- verwendet werden:

```
object.name <- 3*4
```

Auch andere Werte wie Worte können in Objekten gespeichert werden:

```
object.name.2 <- "Hochschulabschluss"
```

Zur Inspektion des Objekts kann der Name des Objekts eingegeben werden:

```
object.name
## [1] 12
```

(Wickham & Grolemund, 2016, Kap. 4.2)

Neue Objekte in R erstellen (II)

→ Objekte werden unter *Environment* im rechten oberen Fenster in RStudio angezeigt.

→ Existierende Objekte werden überschrieben, nicht vorhandene Objekte werden neu erzeugt.

CAVE

- Objektnamen müssen mit einem Buchstaben beginnen und können nur Buchstaben, Zahlen, Unterstriche und Punkte beinhalten.

→ Wir empfehlen, objektnamen.mit.punkten.zu.trennen und immer klein zu schreiben

(Wickham & Grolemund, 2016, Kap. 4.2)

Objektklassen

Objektklassen beschreiben unterschiedliche Typen von Daten

- **numerics/ doubles:** in Zahlen gespeicherte Daten (z.B. Alter)
- **characters:** in Worten gespeicherte Daten
- **logicals:** binäre Variablen, die anzeigen, ob eine Bedingung “TRUE” oder “FALSE” ist
- **factors:** in Zahlen gespeicherte Daten, wobei jede Zahl ein anderes Level einer Variable anzeigt (z.B. 1 = “wenig,” 2 = “mittel,” 3 = “hoch”)

CAVE

Die Klasse einer Variable hat Implikationen auf weitere Analyseschritte. Für characters kann z.B. kein Mittelwert berechnet werden.

(Harrer et al., 2021, Kap. 2.5)

Welche Variablen sind welchen Objektklassen zugeordnet?

Alle Variablen des Datensatzes: Funktion glimpse aus dem package tidyverse

```
library(tidyverse)
glimpse(data)
## Rows: 264
## Columns: 34
## $ id            <chr> "stress_gui_002", "stress_gui_140", ~
## $ group         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ~
## $ sex           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## ...
```

Einzelne Variablen: Funktion class aus dem R base package

```
class(data$id)
## [1] "character"
```

(Harrer et al., 2021, Kap. 2.5)

Operatoren

Überblick Operatoren

✓ Einige Operatoren haben Sie bereits kennengelernt

- **Grundrechenarten:** +, -, *, /
- **Potenz:** ^2
- **Zuweisungsoperator:** <-
- **Vergleichsoperatoren:** >, >=, <, <=, != (nicht gleich), == (gleich)
- $\rightarrow$ Mit Vergleichsoperatoren kann getestet werden, ob Vergleiche zweier Zahlen wahr (TRUE) oder falsch (FALSE) sind
- **Boole'sche Operatoren:** & (und), | (oder), ! (nicht)

(Wickham & Grolemund, 2016, Kap. 5.2)

Praxis-Teil:

Fehlermeldungen

Überblick Fehlermeldungen (I)

Fehlerarten

- **Errors**: Eine Funktion kann nicht ausgeführt und muss gestoppt werden.
- **Warnings**: Es ist ein Fehler aufgetreten, aber die Funktion kann trotzdem (teilweise) ausgeführt werden.
- **Messages**: Information, dass eine Aktion für den Benutzer/ die Benutzerin ausgeführt wurde.

Don't panic

- Fehlermeldungen in R sind am Anfang meist sehr verwirrend

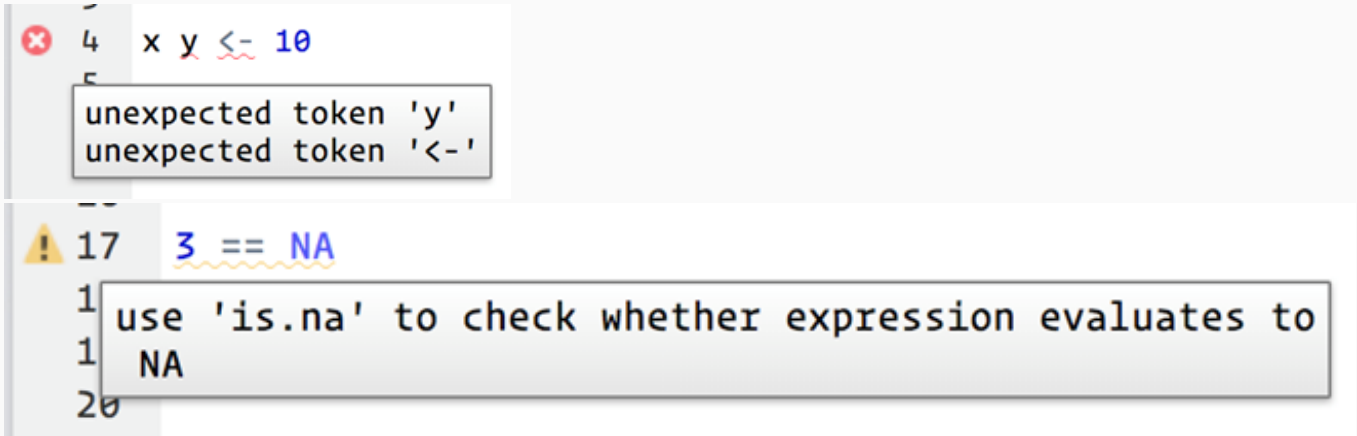
→ Im Laufe der Zeit werden Fehlermeldungen immer informativer und leichter zu entziffern

→ Googlen hilft immer :)

(Wickham & Grolemund, 2016, Kap. 6.2)

Good news: Im Skript in RStudio **werden mögliche Fehler häufig automatisch markiert**

Syntaxfehler werden mit einem roten Kreuz und potentielle Probleme mit gelbem Ausrufezeichen am linken Rand markiert:



(Wickham & Grolemund, 2016, Kap. 6.2)

Referenzen

Referenzen i

Harrer, M., Cuijpers, P., A, F. T., & Ebert, D. D. (2021). *Doing meta-analysis with R: A hands-on guide* (1st ed.). Chapman & Hall/CRC Press.

Wickham, H. (2019). *Advanced r*. chapman; hall/CRC.

Wickham, H., & Grolemund, G. (2016). *R for data science: Import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.".