

R-Kurs: Tag 1

Lukas Mödl, Matthias Becher, Erin Sprünken

Institut für Biometrie und Klinische Epidemiologie

Charité - Universitätsmedizin Berlin, Berlin

erin-dirk.spruenken@charite.de

December 15, 2021





- 1 Einführung
- 2 Wie spreche ich mit R?
- 3 Daten
- 4 Speichern von Daten

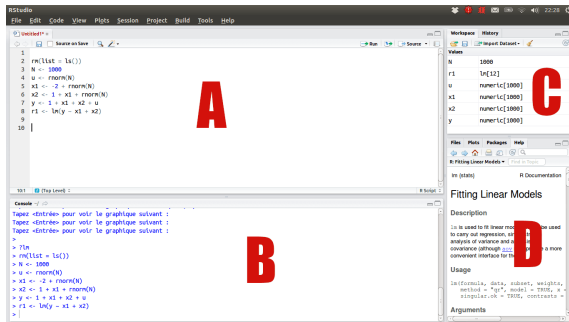
Motivation

- ▶ Statistische Analysen
- ▶ Handlungsspielraum verglichen mit SPSS, SAS, Stata
- ▶ Open Source

Installation

- ▶ R (Software) 
- ▶ RStudio (GUI = Graphical User Interface = Arbeitsumgebung)  Studio
- ▶ Alternativ (aber von uns nicht empfohlen): xcode, Visual Studio, Texteditor

Oberfläche von RStudio



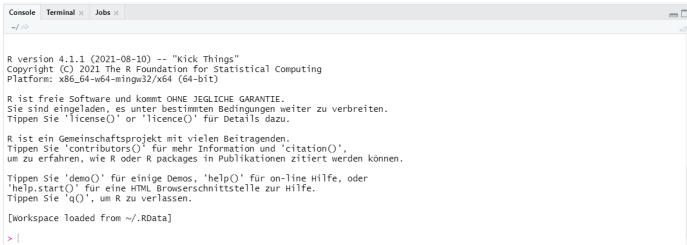
A Skriptfenster

B Konsole / Eingabebezeile

C Workspace, Daten

D Plots, Hilfe, Dateibrowser, Paketmanager

Eingabezeile und Taschenrechner



```
Console Terminal Jobs  
~/>  
  
R version 4.1.1 (2021-08-10) -- "Kick Things"  
Copyright (C) 2021 The R Foundation for Statistical Computing  
Platform: x86_64-w64-mingw32/x64 (64-bit)  
  
R ist freie Software und kommt OHNE JEGLICHE GARANTIE.  
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.  
Tippen Sie 'license()' or 'licence()' für Details dazu.  
  
R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.  
Tippen Sie 'contributors()' für mehr Information und 'citation()',  
um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.  
  
Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder  
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.  
Tippen Sie 'q()', um R zu verlassen.  
  
[workspace loaded from ~/.RData]  
> |
```

- ▶ Nur „Console“ relevant
- ▶ In der Konsole kann man Befehle eintippen und mit Enter ausführen
- ▶ Beispiel Taschenrechner:
 - ▶ $3 + 2$
 - ▶ $5 * 4$
 - ▶ $20 - 8$
 - ▶ $12 / 16$

Fehler, Warnungen, NA, NaN, Inf

Was passiert, wenn wir `5!` eintippen?

R gibt uns einen Fehler zurück. Es ist wichtig, dass wir wissen, wie wir R einen Befehl mitteilen.

Also: `factorial(5)`.

- ▶ $3 / 0 \Rightarrow \text{Inf}$; R löst die Aufgabe numerisch und landet bei einem Grenzwert (hier $\text{Inf} = \infty$).
- ▶ $0/0 \Rightarrow \text{NaN}$; Steht für „Not a Number“
- ▶ NA steht einfach für einen fehlenden Wert. Dies tritt vor allem auf, wenn bei einer Operation nicht angegeben wurde, wie mit vorhandenen fehlenden Werten umgegangen werden soll.

Mathematische Ausdrücke

▷ `factorial(x)` = $x!$

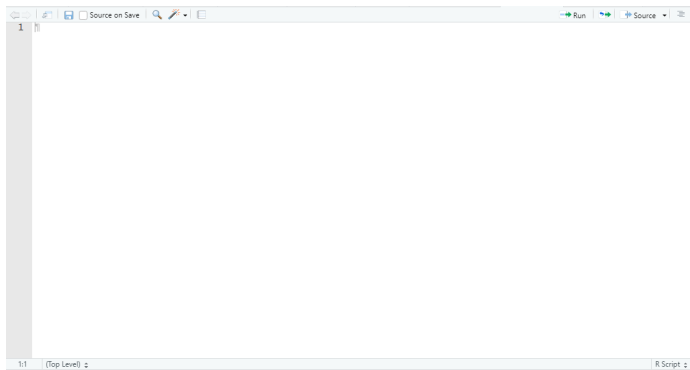
▷ `exp(x)` = e^x

▷ `log(x)` = $\log(x)$

▷ `sqrt(x)` = \sqrt{x}

▷ `abs(x)` = $|x|$

Skriptfenster



- ▶ Es ist umständlich, Befehl für Befehl in die Konsole zu schreiben
- ▶ Im Skriptfenster können wir beliebig viele Befehle nacheinander reinschreiben und ganze Blöcke auf einmal ausführen
- ▶ Technisch passiert beim Ausführen eines Skripts nichts anderes, als dass Zeile für Zeile an die Konsole übergeben wird

Variablen zuweisen

Oft wollen wir Ergebnisse oder Daten irgendwo speichern, um sie später weiterverwenden zu können. Dazu müssen wir Variablen zuweisen. In R wird dazu (historisch) ein Pfeil verwendet: `<-`. Genauso funktioniert allerdings das `=` um Daten zuzuweisen. Variablen sind „Case Sensitive“, d.h. Groß- und Kleinschreibung ist wichtig!

▶ `x <- 3.1415`

▶ `b <- 4`

▶ `B = 5`

▶ `d <- c(1,2,5,7)`

Sonstiges

R bietet einige nützliche Funktionen. Zwei besonders nützliche sind `?` und `rm(x)`.

- ▶ Wenn wir wissen wollen, was eine Funktion macht, können wir mit dem Fragezeichen gefolgt vom Namen der Funktion ihre Dokumentation aufrufen. Z.B.: `?rm`
- ▶ Wenn wir das nun eingeben, erfahren wir direkt mehr über die zweite nützliche Funktion: Wir können in die Klammern Objekte reinschreiben, die wir löschen wollen. Von der vorherigen Folie haben wir noch `x`, `b`, `B` und `d`
- ▶ Wir wollen nun `B` und `b` löschen, also: `rm(B, b)`

Datentypen

Welche Arten von Daten gibt es in R?

- ▶ Numeric \Leftrightarrow Zahlen, `is.numeric()`
- ▶ Character \Leftrightarrow Buchstaben/Wörter `is.character()`
- ▶ Factor \Leftrightarrow Kategorische Variablen `is.factor()`
- ▶ Date \Leftrightarrow Datum/Zeit `is.Date()`

Vektor

R ist eine Vektor-Sprache: Fast alle Datenkonstruktionen sind Formen oder Erweiterungen eines Vektors.

Ein R-Vektor kann nur einen Datentypen beinhalten.

- ▶ `c()`
- ▶ `seq()`
- ▶ `rep()`
- ▶ `is.vector()`

Matrix

Eine Matrix ist eine Aneinanderreihung von Vektoren (nebeneinander, untereinander).
Eine R-Matrix kann nur einen Datentypen beinhalten.

- ▶ `matrix()`
- ▶ `cbind()`
- ▶ `rbind()`
- ▶ `is.matrix()`

Rechenoperationen

Man kann in R einfach mit Vektoren und Matrizen rechnen. Grundsätzlich: Elementweise! Was erhält man bei folgendem Befehl?

```
c(3.1415, 5, 1, 2/3) * seq(1, 8, 2)
```

R rechnet folgendermaßen:

$$\begin{pmatrix} 3.1415 \\ 5 \\ 1 \\ \frac{2}{3} \end{pmatrix} * \underbrace{\begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}}_{\text{seq}(1, 8, 2) = \text{c}(1, 3, 5, 7)} = \begin{pmatrix} 3.1415 \cdot 1 \\ 5 \cdot 3 \\ 1 \cdot 5 \\ \frac{2}{3} \cdot 7 \end{pmatrix}$$

Liste

Was, wenn wir verschiedene Datentypen in einem Vektor speichern wollen?

Antwort: Liste.

Listen sind etwas abstraktere Varianten des klassischen Vektors, da Listenelemente nicht den gleichen Datentyp haben müssen. Listen sind flexibel und können sogar verschachtelt werden, also eine Liste kann weitere Listen enthalten.

▶ `list()`

▶ `c()`

▶ `$`

Data Frame

Listen sind manchmal etwas unübersichtlich. Eine spezielle Form einer Liste ist der Data Frame. Optisch sieht der Data Frame aus wie eine Matrix, allerdings können hier unterschiedliche Datentypen gespeichert werden. Unterschiedliche Spalten (nur Spalten, keine Zeilen) können vom unterschiedlichen Typ sein.

- ▶ `data.frame()`
- ▶ `rbind()`
- ▶ `cbind()`
- ▶ `$`

Speichern

Oft kommt es vor, dass wir nicht nur einmal an etwas arbeiten. Es macht also Sinn, Daten zu speichern. Unter anderem bietet R einen Dateityp an, mit dem man R-Daten speichern kann:

*.RData

- ▶ `save()`
- ▶ `write.table()`
- ▶ `write.csv()`